(12)

Technical Report 1300

Pose Determination of a Grasped Object Using Limited Sensing

David M. Siegel

MIT Artificial Intelligence Laboratory

1998

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1991	technical report

4. TITLE AND SUBTITLE
Pose Determination of a Grasped Object Using Limited Sensing

5. FUNDING NUMBERS
N00014-86-K-0685

6. AUTHOR(S)
David M. Siegel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER
AI-TR 1300

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Information Systems
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION AVAILABILITY STATEMENT
Distribution of this document is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstract. This report explores methods for determining the pose of a grasped object using only limited sensor information. The problem of pose determination is to find the position of an object relative to the hand. The information is useful when grasped objects are being manipulated. The problem is hard because of the large space of grasp configurations and the large amount of uncertainty inherent in dexterous hand control. By studying limited sensing approaches, the problem's inherent constraints can be better understood. This understanding helps to show how additional sensor data can be used to make recognition methods more effective and robust.

This report introduces three approaches for pose determination. The first is based on an interpretation tree representation of possible object feature placements on finger segments. The tree is built in real-time based on the hand's configuration and an object model. The method is highly efficient as it only explores consistent paths through the tree.

(continued on back)

14. SUBJECT TERMS (key words)		15. NUMBER OF PAGES
robotics	haptics	179
recognition	pose determination	16. PRICE CODE
hands		

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Block 13 continued:

The second approach is based on storing past recognition experiences in a memory. Recognition becomes a fast lookup operation. The number of possible grasps stored in the memory can be kept small by exploiting a grasp acquisition strategy constraint. The memory can be further compacted by using interpolation schemes. Various approaches for organizing, filling, and using the recognition memory are investigated.

The third approach explores how additional contact sensing information can be used for recognition. Fingertip force sensors are used to measure contact surface normals. An object's pose can be refined by fitting these normals to a model of the object. Since contact sensors produce local readings, and since the fitting process requires global information, calibration becomes an important issue. The method explored provides a way to self calibrate for sensor orientation errors.

An important theme throughout the report is that knowledge of the strategy the hand uses to acquire an object provides a very important recognition constraint. While a dexterous hand has a very large configuration space, the number of unique grasps that correspond to a given strategy are limited. Knowledge of the grasp acquisition strategy makes pose determination easier.

Pose Determination of a Grasped Object Using Limited Sensing

by

David Mark Siegel

Submitted to the Department of
Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

May 1991

Pose Determination of a Grasped Object Using Limited Sensing

by

David Mark Siegel

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Abstract. This report explores methods for determining the pose of a grasped object using only limited sensor information. The problem of pose determination is to find the position of an object relative to the hand. The information is useful when grasped objects are being manipulated. The problem is hard because of the large space of grasp configurations and the large amount of uncertainty inherent in dexterous hand control. By studying limited sensing approaches, the problem's inherent constraints can be better understood. This understanding helps to show how additional sensor data can be used to make recognition methods more effective and robust.

This report introduces three approaches for pose determination. The first is based on an interpretation tree representation of possible object feature placements on finger segments. The tree is built in real-time based on the hand's configuration and an object model. The method is highly efficient as it only explores consistent paths through the tree.

The second approach is based on storing past recognition experiences in a memory. Recognition becomes a fast lookup operation. The number of possible grasps stored in the memory can be kept small by exploiting a grasp acquisition strategy constraint. The memory can be further compacted by using interpolation schemes. Various approaches for organizing, filling, and using the recognition memory are investigated.

The third approach explores how additional contact sensing information can be used for recognition. Fingertip force sensors are used to measure contact surface normals. An object's pose can be refined by fitting these normals to a model of the object. Since contact sensors produce local readings, and since the fitting process requires global information, calibration becomes an important issue. The method explored provides a way to self calibrate for sensor orientation errors.

An important theme throughout the report is that knowledge of the strategy the hand uses to acquire an object provides a very important recognition constraint. While a dexterous hand has a very large configuration space, the number of unique grasps that correspond to a given strategy are limited. Knowledge of the grasp acquisition strategy makes pose determination easier.

Thesis Supervisor: Tomás Lozano-Pérez
Professor of Electrical Engineering and Computer Science

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided by the Office of Naval Research under University Research Initiative contract N00014-86-K-0685. Support for the author was provided, in part, by fellowships from the National Science Foundation and from the International Business Machines Corporation.

To my Mother, my Father, my Sister Sharon,
and my Grandmother Helen

Acknowledgments

I'd like to thank my advisors Tomás Lozano-Pérez, Eric Grimson, and Ken Salisbury for their help, encouragement, and support throughout my studies at the MIT AI Lab. I also deeply appreciate the help and support I have received from John Hollerbach. The lab's directory, Patrick Winston, the faculty, and the students have all contributed to making my work here both possible and highly enjoyable.

Much of the experimental work that I conducted was made possible by the efforts of a large group of people. John Hollarbach, along with Steve Jacobsen at the University of Utah, made the Utah-MIT hand a reality. I thank Ken Salisbury for his robotic hand, and for his efforts in developing and porting its control software. Dave Brock's fingertip sensors, and all his help on using them, proved indispensable. I thank Thomas Massie for his help with the planar hand.

Nancy Pollard helped me in endless ways throughout my studies at MIT. She collaborated with me on much of the work described in Chapter 5. Her helpful comments on drafts of this thesis, and on related papers, are much appreciated. Not only is Nancy a very talented researcher, but she is a wonderful friend.

Sundar Narasimhan and I have worked together our entire stay at MIT. He's a close friend, a great hacker, and I hope to continue my work with him throughout my career.

Laurel Simmons has always been a very special person. Not only has Laurel kept this place running — she's kept the lab, well, the AI Lab! I'm happy to have been able to help her out a little. We've wired networks, built sensors, and a whole lot more. She's made this place far more fun for me than it otherwise would have been. One of the hardest parts about leaving MIT is realizing that I won't be seeing her often enough.

Many other people have provided deeply appreciated assistance, including Chris Atkeson, Steve Drucker, Iñaki Garabieta, Chris Lindblad, Patrick O'Donnell, Jose Robles, Jerry Roylance, and Ron Wiken.

Many of my friends at the lab helped with my work, and made a lighter side to life at MIT, which is necessary for getting through this place. I thank Mike Caine, Camille Chammas, Nomi Harris, Ray Hirschfeld, David Michael, Barb Moore, Karen Sarachik, and Patrick Sobalvarro.

Last, but most importantly, I thank my family for all their support and love.

Contents

List of Figures

List of Tables

Introduction

Chapter 1

We are certainly very good at performing manipulation with our hands. Along with a set of tools, we can make our hands perform an enormous variety of tasks. What makes them so versatile? A combination of factors are probably at play. Our fingers can move quickly and accurately. They can perform both delicate operations or exert powerful grasping forces. Our touch sensors provide a wealth of information that undoubtedly helps make them so useful.

Dexterous robot hands will ultimately give machines some of the capabilities that our own hands have, helping to make them more useful. A hand is most appropriate for performing manipulations in *unstructured* environments, particularly where it is undesirable for humans to work. An example of such a task is the cleanup of hazardous sites. The work required to contain the tragic accident at Chernobyl needlessly exposed thousands of people to deadly radiation. Potentially, mobile robots equipped with dexterous hands could be used in similar situations.

This report addresses a small part of the overall problem of giving a robotic hand human level dexterity, the problem of determining the pose of a grasped object.

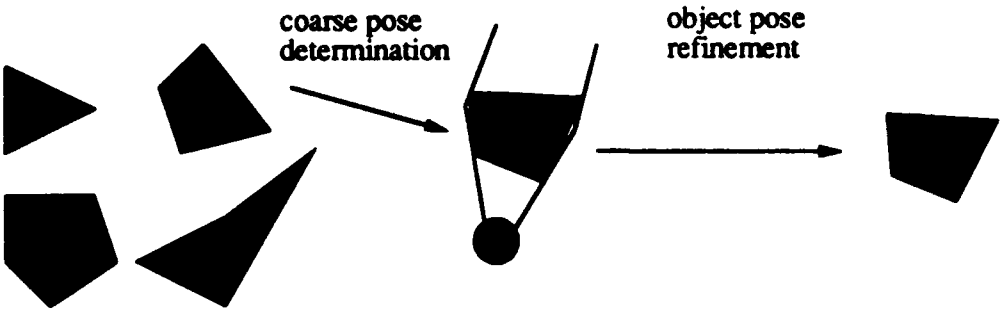


Figure 1.1: *Pose determination. The relationship between recognition, determination, and refinement.*

1.1 Pose Determination

The class of problems studied in this report can be stated simply: Given a hand grasping an object, and given models of a small number of objects, determine the object, its position, and its orientation (see Figure 1.1). These problems are often referred to as small set recognition and pose determination. Recognition and determination are related problems, and it is sensible to study them together. For convenience, in this report the term *pose determination* often refers to these problems collectively.

1.1.1 Relevance of the Studied Problem

Why is the pose determination problem interesting? A hand is used to grasp an object in order to manipulate the object. The manipulation can be simple, such as moving the object from one location to another, or it can be complex, such as grasping a tool for performing an assembly operation. In either case, knowledge of the location of the object within the hand is useful for insuring that the manipulation is performed correctly. While certain manipulation strategies might not require pose information, it is not hard to imagine that the information would be beneficial.

The problem is hard because of the large space of grasp configurations and the large

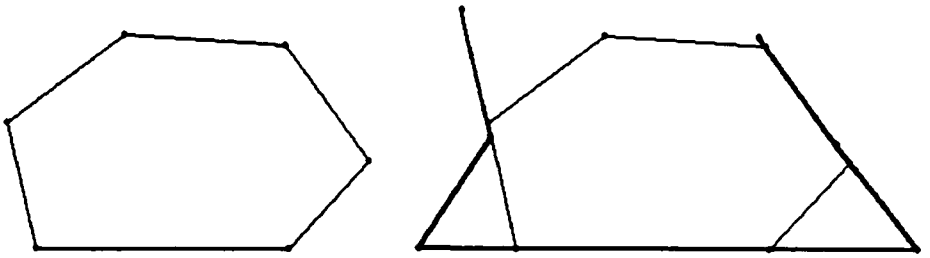


Figure 1.2: *Object and its grasp. The object, shown on the left, is grasped by a hand, shown on the right. For pose determination, only the hand shape is known. The one or more positions of the object that are consistent with that hand shape are desired.*

amount of uncertainly inherent in dexterous hand control. By studying limited sensing approaches, the problem's inherent constraints can be better understood. This understanding helps to show how additional sensor data can be used to make determination methods more effective and robust.

1.1.2 Example of the Studied Problem

This section describes a typical pose determination problem. The inputs to the problem include a model of the robotic hand, a model of the object, and the hand's joint angle positions after the object has been grasped. The object is shown in Figure 1.2, on the left. A grasp of the object is shown on the right. For pose determination, the hand shape is known, while the object position is unknown, and is to be found. Figure 1.3 shows the hand shape, along with the poses that are consistent with the shape. The only sensor information used for this determination is the set of joint angle readings from the hand. No additional sensor data is required to obtain the set of solutions that are shown in the figure. In this case, since the solution set contains the object's actual pose along with several other consistent candidates, additional sensor data would be required for unambiguous determination. The first row of candidate poses, for example, could be ruled out if information about the joint torque readings were available. If the joints are known to be curled as far forward as possible, the links must make the necessary contacts

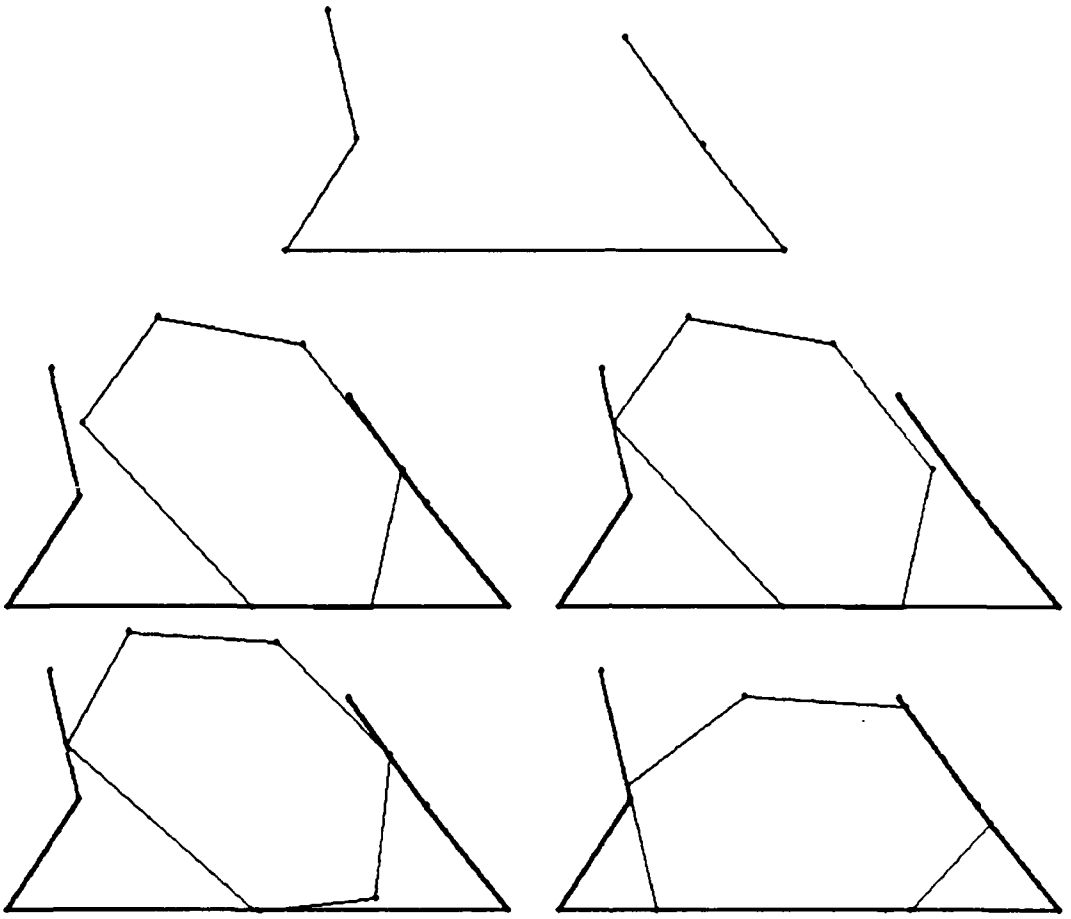


Figure 1.3: *Hand shape and potential poses. The object poses shown here are consistent with the hand's shape.*

with the object to constrain the finger motion. The second row of poses shown in the figure are entirely consistent with both the geometric constraints and the joint torque data. Additional sensor information would be needed to distinguish between them.

1.1.3 Potential Information Sources

The information available for pose determination can be grouped into that from the constraints inherent in the problem and that from sensors. An understanding of a problem's constraints often leads to powerful techniques for its solution. An understanding of how

the constraints can be used to solve a problem helps to interpret sensor data. With this in mind, this report explores the pose determination problem by relying heavily on the problem's constraints, and using little external sensor data. This section explores the potential information sources available, and explains why the ones that are used in this report were selected.

Geometric Constraints

Geometric constraints provide a powerful inherent source of information for determination. The methods described in this report exploit both the geometry of the objects and the shape of the hand.

Another inherent constraint is provided by the grasping strategy. A particular grasping strategy limits the possible hand shapes that can occur. As will be seen, this particular constraint can be directly exploited for determination.

Contact Sensors

A variety of sensors can be used to provide the raw data for pose determination. Perhaps the most basic is kinesthetic (joint position) sensing. This type of information is readily available on dexterous hands, as joint position sensing is usually required for their low level servo control. Cutaneous (tactile) sensing is more advanced, from a hardware standpoint. The term *haptics* is often used to refer to these hand-based senses. Visual sensing can also be used, but as will be seen, it is less desirable than haptics for many applications.

Tactile sensors give a small window of information at their contact point, which provides data suitable for local methods. Curvature detection is an example of a local problem that can use tactile sensor data. Kinesthetic sensing can be used to obtain the hand's shape, which provides data suitable for global methods. Pose determination is an example of a global problem that can use hand shape data.

Vision Sensors

Unlike tactile sensors, vision can give a large window of information. With a camera in the proper location, one view can provide far more information than could be obtained even by multiple probes with tactile sensor. Thus, vision potentially provides an easy way to acquire global features.

There are a number of reasons why vision is not a particularly good source of information for pose determination. The most significant problem is the potential occlusion of the grasped object by the hand. For relatively small objects the hand can obscure many, perhaps all, of the object's visible features. In addition, discrimination between features of the hand and features of the object becomes difficult as more of the object is occluded.

Typically, the pose of an object with respect to the position of the hand is required. Cutaneous and kinesthetic sensing take readings in the hand frame, which is desirable. Readings from a vision system would usually be made from a different frame, requiring additional calibration.

The resolution of vision systems may not be adequate for tasks that require extremely accurate pose determination. The advantage that vision offers, that of being a global sense, works against the requirement for precision. Since haptic information sources take readings at the site of contact between the hand and the object, they can potentially give better readings than a remote vision system could.

Vision is a useful sense for determining an object's position prior to being grasped. Unfortunately, when the object is touched, it usually moves in a hard to predict way. Because of this, the pre-grasp position does not always give a good indication of the object's final pose.

Information Used by the Studied Approaches

The pose determination algorithms presented in this report use only the inherent problem's constraints, along with the addition information provided by joint angle and joint torque sensors. As explained above, the motivation for this limited information approach is twofold. A full exploitation of the basic constraints inherent in the problem leads to a better understanding of the problem. Methods that can work on kinesthetic information will facilitate this understanding. From a practical standpoint, kinesthetic information is the most readily available data source today. This makes it even more desirable to develop methods that work with this type of data.

It is important to note that while the techniques explored in this report rely only on minimal sensor data, they can easily incorporate, and will benefit from, the additional information that tactile sensors provide. The constraints that are exploited are useful independent of the external sensor data available. For example, more data can be used to reduce ambiguity and to provide redundancy that would make the methods more robust.

1.1.4 Approaches

This report introduces three approaches for pose determination, as diagrammed in Figure 1.4. The first approach is constraint-based, and uses an interpretation tree representation of possible object feature placements on finger segments. The tree is built in real-time based on the hand's configuration and an object model. The method is highly efficient as it only explores consistent paths through the tree.

The second approach is memory-based, and uses past experiences for determination. Determination becomes a fast lookup operation. Possible grasps can be kept sparse by exploiting a grasp acquisition strategy constraint. The memory can be compacted using interpolation schemes. Various approaches for organizing, filling, and using the determination memory are investigated.

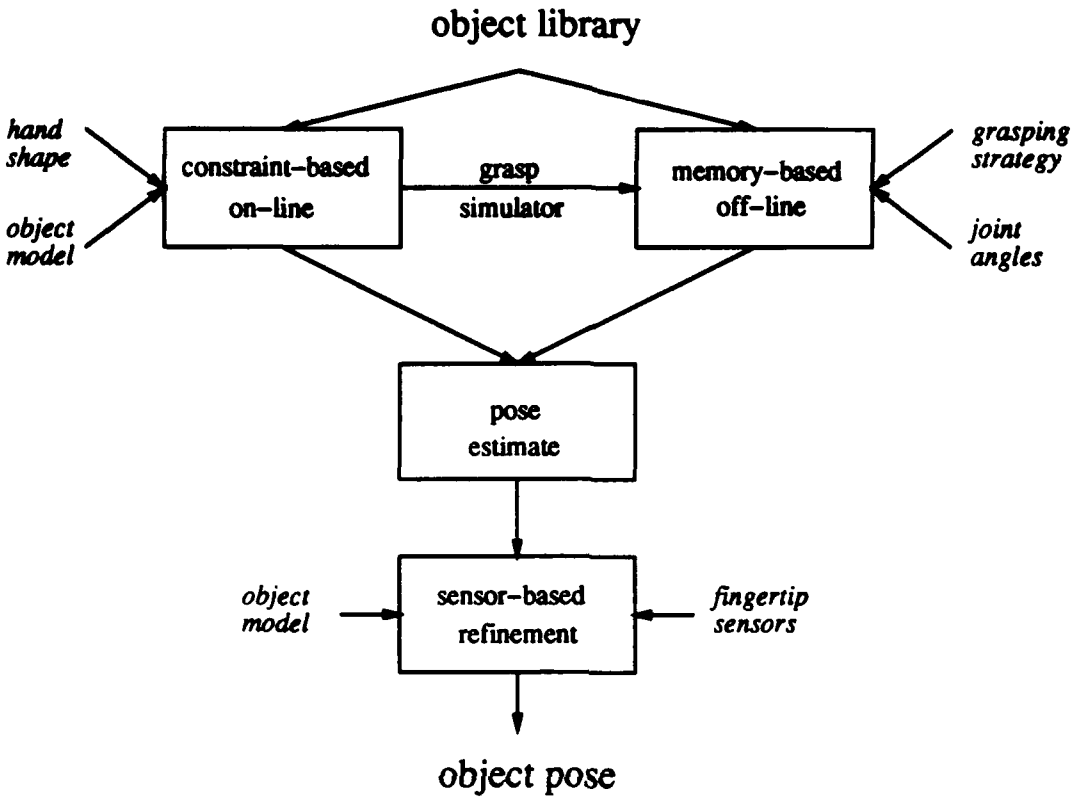


Figure 1.4: *Algorithms and their relationship. The algorithms studied in this report , and their relationship, are diagrammed in this figure.*

The third approach is sensor-based, and explores how additional information can be used for determination. Fingertip force sensors are used to find contact surface normals. An object's pose can be refined by fitting these normals to a model of the object. Since contact sensors produce local readings, and since the fitting process requires global information, calibration becomes an important issue. The method explored provides a way to self calibrate for sensor orientation errors.

As will be seen, the approaches studied are straightforward yet powerful. This work shows that *simple* methods, using basic constraints and limited sensor data can solve a *wide* class of pose determination problems. Showing this to be true is the motivation for this work.

1.2 Grasp Planning and Object Acquisition

In this report , a distinction is made between objects that have been grasped using a plan, and objects that have been acquired without a plan. Typically, a *grasp planner* is given as input the position of the object to be grasped. The plan generated has a certain tolerance to error in object placement, though undoubtedly the grasp will fail if the object's actual position is far from the modeled position. The error tolerance of a particular grasp is often a factor that is considered in the planning process. A *grasp acquirer* works without knowledge of an object's position. Instead, a generic hand motion, perhaps modified by sensor feedback, is used to acquire the object.

There are many examples of problems that are suitable for planned grasps. In particular, when the world is well known, or when external sensors like vision can identify the location of objects, the use of a planner is appropriate. Since planning can be a slow process, it is helpful if the world is relatively static. Certainly, if it is changing faster than a planner can plan, there will be problems. As uncertainty in the world increases, planning becomes more difficult. Planners can be designed to handle uncertainty up to a point (see Mason [70]). In the limiting case, where nothing specific is known about the world, there is little reason to plan.

Grasp acquisition is useful for problems where planning is inappropriate. This include when using a planner may be difficult, because there is too much uncertainty, or when it may be impossible, because there is no information at all. Haptic exploration is an important class of motions where planning is not appropriate. Reaching into a bin, identifying a particular item, and grasping it, is an example of such a motion. A more practical example includes tool retrieval. NASA is interested in a device to capture free-floating tools and other small objects from space. An astronaut might accidentally release a tool during a space walk, creating a flying obstacle that could collide with a spacecraft. It is unlikely that planning a grasp to retrieve the floating object would be appropriate. Using an acquisition strategy designed to capture the class of objects that

can potentially be released seems more reasonable.

The pose determination strategies studied in this report are useful for finding the position of an object that as been acquired, where no a priori knowledge of the object's position is available. The methods are also suitable for distinguishing among a small set of objects that are being acquired. An easy way to do this is to simply invoke the pose determination strategy once for each object in the set. Ideally, the strategy will find a pose only for the correct object. Finding multiple objects implies that the information being used for pose determination is just not enough to distinguish among the objects.

Pose determination is also useful for verifying that a planned grasp has been executed correctly. Because of errors, a planned grasp is never executed exactly as intended. Verification of the plan to insure that it has completed satisfactorily is a useful step to perform before proceeding to the next manipulation. Ideally, the planned grasp would be executed in a closed loop, where errors are detected in real-time, and recovery strategies are part of the plan. Post-execution verification is useful, though not necessarily the best approach.

1.3 Hand Design and Shape Information

The design of a hand directly affects its haptic information content. For example, hands with more links can better conform to the shape of an object, providing more clues as to the object's shape. Double jointed hands provide more shape information than a single jointed hand, as shown in Figure 1.5. Some of these issues will be examined in subsequent chapters of this report .

1.4 Assumptions

The work in this report makes certain assumptions about the world. The methods that are described are not, in all cases, limited by these assumptions. Rather, the methods have been tailored to best fit the natural constraints that these assumptions provide.

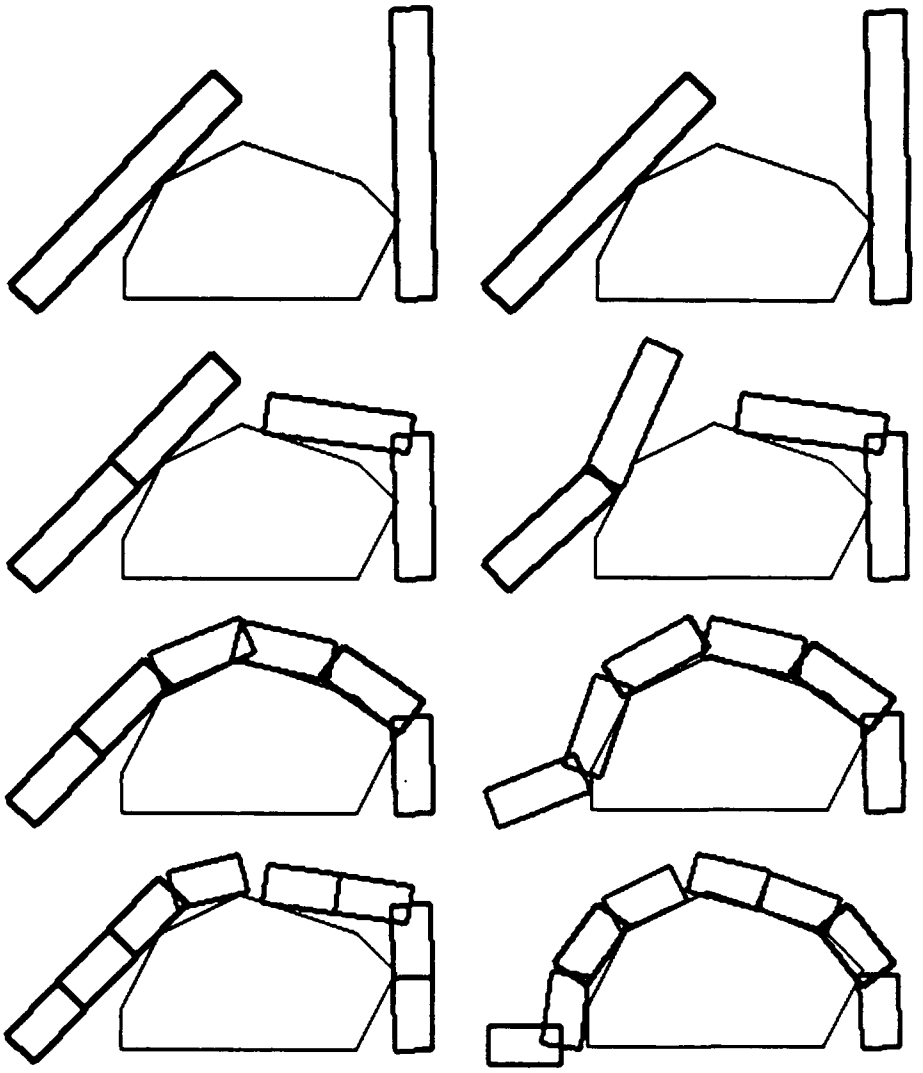


Figure 1.5: *Double joint grasps. This figure compares grasps using a double joint hand with a single jointed hand. The double jointed grasps are shown on the right.*

As each method is presented, its limitations and its best applications are discussed.

The objects in the world are assumed to be polyhedral. This permits simpler models and algorithms to be used compared to objects represented with a more general scheme. In many cases, the ideas presented can be extended to a fully general representation. In some cases they cannot. Each chapter discusses this issue in some detail.

By assuming that objects are resting on a table-top before they are grasped, it is

possible to use two dimensional methods to solve certain three dimensional problems. This *table-top* constraint reduces the possible configuration space of the object from three dimensions to n spaces in two dimensions, where n is the number of object faces. This assumes that an object can only rest on one of its stable faces. For an uncluttered workspace, this is probably a reasonable assumption. The pose determination methods described in this report are implemented in two dimensions, and can be directly used for three dimensional problems using this added constraint. It is important to note that full three dimensional extensions to the methods are possible, and are explored in some detail.

One additional assumption made is that the world contains invariants that can be exploited by determination schemes. Object models are, of course, invariant. Their shape, compared with a hand's shape, provides important clues that can be used for pose determination. Chapter 5 exploits the invariant of gravity. Gravity provides a force that can be used to relate *different coordinate systems* together, and is used to refine an initial estimate of an object's pose to a more precise estimate.

1.5 Contributions

The most important contributions of this work can be summarized as follows:

- It is shown that a hand's shape often provides enough information to uniquely determine the pose of a grasped object.
- It is shown that knowledge of the grasp acquisition strategy provides a useful recognition constraint.
- An efficient algorithm for finding object poses based on tree pruning is presented.
- A fast algorithm for finding object poses based on experiences stored in a memory is presented.

- Minimal sensing that determines the links that are in contact with an object is shown to be very effective for reducing ambiguity in the results.
- Experimental results explore the tradeoff in recognition power obtained by adding links or sensors to a hand.
- A method for refining contact surface normals from fingertip sensors is presented. These contact normals can be used for pose determination.
- Experiments indicate that it is hard to extract global information from the local measurements obtained from fingertip sensors.

The methods that are explored in this report are not complex. In fact, their *simplicity* can be considered an important contribution in itself. This report will show that pose determination can be accomplished with simple methods and minimal sensing. While this does not imply that more complex methods are unnecessary, it does indicate that the constraints that are used lend themselves to straightforward determination algorithms.

1.6 Overview of the Report

Before describing the pose determination and refinement algorithms, Chapter 2 overviews hands, sensing, grasping, and recognition. The chapter provides a general review of research related to pose determination. Each subsequent chapter mentions work more directly related to the particular algorithms being presented. Following the review, the next three chapters are organized around the diagram shown in Figure 1.4. Chapter 3 describes a constraint-based determination method. Chapter 4 describes a memory-based determination method. The constraint-based method can be used as the grasp simulator that is required for populating the memory used by the approach in this chapter. These two chapters present methods for finding a pose estimate. Chapter 5 describes a technique for refining pose estimates by using additional sensor data. Finally, conclusions and directions for future research are discussed in Chapter 6.

Hands, Sensors, Grasping and, Recognition

Chapter 2

2.1 Introduction

This chapter reviews relevant research in hands, sensing, grasping and recognition. The topics reviewed cover such a broad area because of their unavoidable interrelationships. As this report attempts to show, it is improper to consider one of these aspects without considering them all. To build a hand for recognition — mechanical design, sensor design, and grasping strategies must all be considered. In each of the next sections, relevant work from these areas is described, with special attention given to the interrelations between them. Specific discussion of how these methods directly relate to the work described in this report is deferred to the related work sections in each of the subsequent chapters.

2.2 Dexterous Hands

Dexterous robotic hands have been studied at least since the early 1960's. Some of the earliest work is by Tomovic [104] and Okada [79]. Other hands have been developed

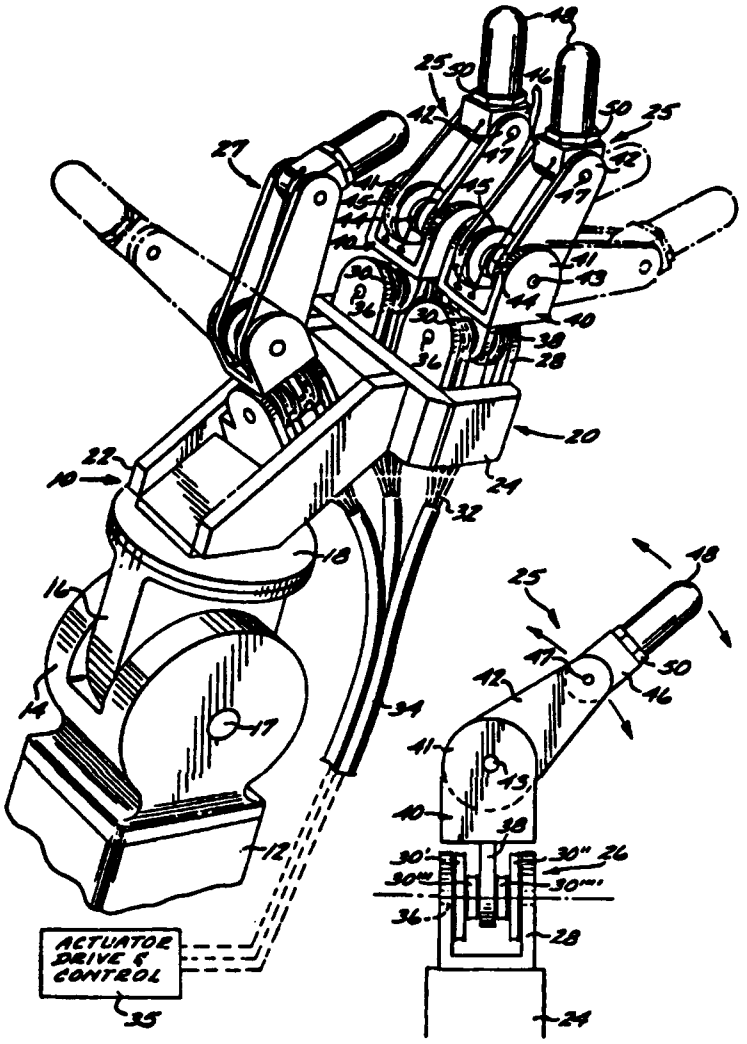


Figure 2.1: *The Salisbury-JPL hand, from Salisbury [90].*

by Salisbury [90], Jacobsen et al. [57], Bologni et al. [14], Abramowitz et al. [1], and Caporali and Shahinpoor [17]. This section discusses a number of these hands in more detail. The section is loosely organized around the motivating design principles used by the researchers.

2.2.1 Mobility Design

A three-fingered hand was designed by Salisbury [90]. In his work, a mobility analysis of various kinematic configurations was performed. The actual design he selected optimized certain criterion according to this consideration. Thus, Salisbury's primary goal was to achieve a mechanical design that was well suited for grasping. The hand is actuated by a servo-motor pack connected to the joints using bicycle-style cables. Figure 2.1 diagrams the configuration that Salisbury selected.

2.2.2 Anthropomorphic Design

Jacobsen et al. [57] developed the Utah-MIT hand based on a belief that an anthropomorphic design has certain inherently desirable traits. The versatility of the human hand is proof that its design is a good one. Using this principle, the four fingered, four jointed hand shown in Figure 2.2 was developed. Special attention was given to the tendon and actuator design. A Kevlar material was used for the tendons, giving them both strength and flexibility. Jacobsen believes that the actuator component is at least as crucial as the mechanics. In the case of the Utah-MIT hand, the pneumatic actuators provide a very human-like natural compliance. Their performance is well suited for grasp acquisition routines, as will be discussed later in this report .

2.2.3 Behavioral Design

While both the Salisbury-JPL and the Utah-MIT hands were designed to have a reasonable kinematic configuration for grasping, their performance was more optimized for mobility, or said another way, dexterity. As an alternative, Greiner's [40] Prehensile Acquisition Linkage Mechanism (PALM) is specifically designed for grasping (Figure 2.3). Mobility is not considered as important. The device has one active and two passive degrees of freedom. The actively controlled tendon can be used to close the hand. The same mechanism will passively curl the hand around objects that are pushed into its

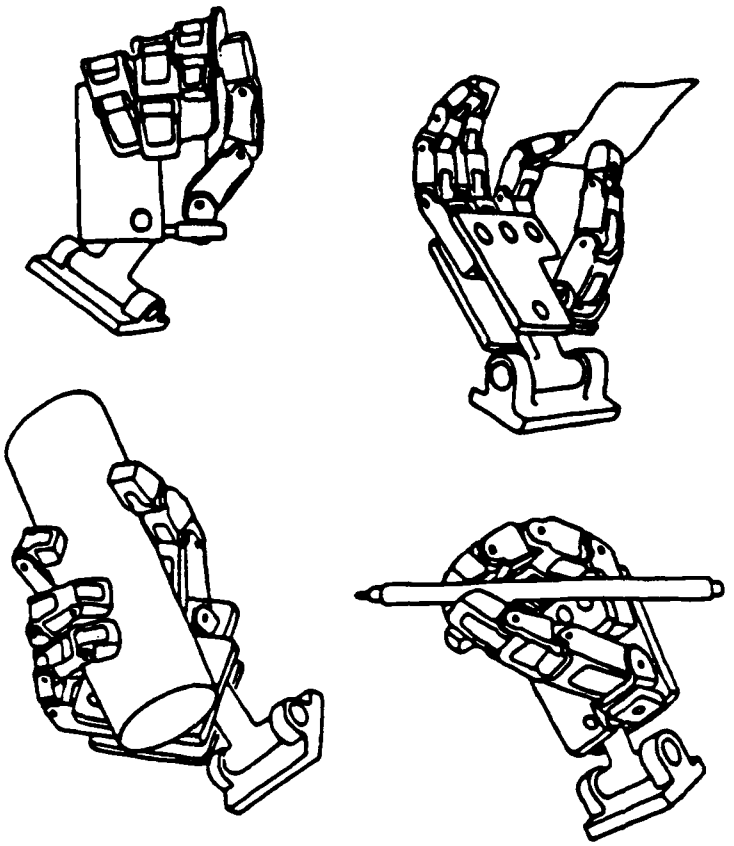


Figure 2.2: *The Utah-MIT hand, from Jacobsen et al. [57].*

links. In essence, the hand implements a grasping strategy in its hardware. As will be seen, the methods described in this report are very suitable for giving a device like the PALM recognition capabilities. The fixed grasping strategy that this hand uses could be exploited by the recognition algorithms.

2.2.4 Other Design Considerations

There are other reasons given for particular hand designs. Hirose and Umetani [49] developed a soft, snake-like grasper. The University of Bologna hand (Bologni et al. [14])

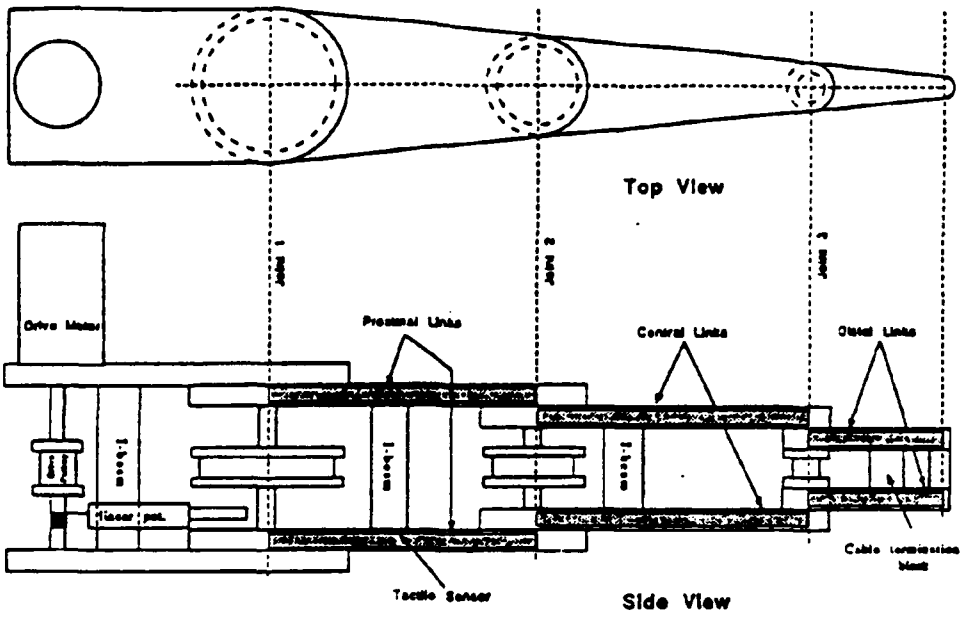


Figure 2.3: *The Greiner PALM, from Greiner [40].*

was designed to be "at least as useful" as a conventional robotic end-effector, while also having "micro-manipulation" capabilities. Abramowitz et al. [1] cited the recent development of tactile sensors as a primary reason for building the Pennsylvania Articulated Mechanical Hand. Sensor equipped hands, he reasoned, are good for three dimensional perception. Rather, it seems likely that hands will be most useful as manipulation devices, where the recognition that they perform is directly related to their manipulation needs. Sensor designs have been motivated by hand designs, not the other way around. Nonetheless, Abramowitz made the good point that hands have an important role in sensing, or haptics. This issue will be explored more later in this section.

Another aspect of hand design that has been considered are the properties of the fingertips themselves. Brockett [16] argued that rheological surfaces are good for grasping. Cutkosky et al. [26] analyzed a number of materials to find their suitability for fingertip surface coverings. The design of a covering becomes more complex when the

fingers are equipped with tactile sensors. Their surface must protect the sensors while not interfering with the transduction process.

From a mechanical standpoint, robotic hands are advanced and are highly dexterous. However, their actuation system are much too bulky. The Utah-MIT hand uses a pneumatic actuation system with flexible Kevlar tendons. While the actuators themselves are fairly compact, a large external air source is required for power. The entire actuator pack is also too bulky for mounting on most robots. Chiarelli and De Rossi [20, 21] and others are studying muscle-like fibers that may be the basis for a far more advanced and compact actuation system of the future.

2.3 Touch Sensors

Touch sensors are thought to be important for many aspects of dexterous hand manipulation. Certainly, an accurate measurement of contact forces is helpful when grasping delicate objects. Feature detection can provide useful information for object recognition and pose determination. Slip detection is helpful for monitoring a grasp to insure that it is being properly maintained. This section reviews tactile sensing, briefly exploring human sensors, the mechanics of sensing, and a variety of robotic devices.

2.3.1 Human Sensors

A goal of tactile sensor designers has always been to duplicate the capabilities of the human sensing system. Human touch sensors have many very desirable properties. Perhaps the most important one of all is that they are compact and reliable. There are thousands of mechanoreceptors in the small confines of the fingers. Humans have four types of touch sensors, each which is specialized for a particular response (Figure 2.4). The Merkel and Ruffini receptors have some static touch response, while the Meissner and Pacinian respond better to a changing stimulation. Human touch sensors can detected pressure, shear, and slip (Johansson et al. [59, 60, 61]). Measurements of the

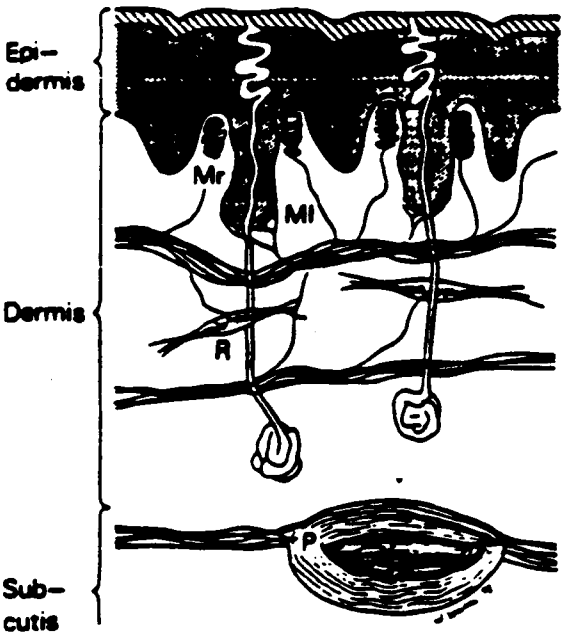


Figure 2.4: *Cross section of human skin.*

performance of the human tactile system made by Vallbo and Johansson [106] indicate that static two point discrimination between 1.5 and 2.2 mm is possible. The sensations that we are so capable of feeling have not yet been entirely duplicated by robotic devices, although recent advances are starting to close the performance gap.

2.3.2 Mechanics of Transduction

All tactile sensors have the common task of detecting some mechanical phenomenon and transducing it to a measurable signal. Ultimately, the signal is converted to digital data for processing by a computer. The earliest tactile sensors used various phenomenon to generate an analog electrical signal. Later devices translated mechanical phenomenon to an optical signal. More recent devices directly convert the sensed phenomenon to a digital signal, bypassing the analog signal stage. See Usher [105] for an overview of sensing, including the physical effects that are available for use in transduction.

The most common property that sensor designers have attempted to detect is pressure. An array of pressure sensors are frequently used to find the contact profiles of objects pushed into them. This would be useful for determining, for example, if an object vertex, edge, or face is making contact with the sensor. Pressure sensors of this type have proven to be the easiest to build.

Other phenomenon that are interesting to detect include shear and torque at the contact point. Measurement of these properties is helpful for determining if slippage is likely to occur. Unlike pressure, shear and torque sensors have been hard to build. There are only a few examples of sensors of this type in the literature.

Another interesting sensor has been used to detect the thermal properties of a material. By applying heat and measuring the resulting temperature gradients, the contact material can often be identified. In addition, slip can be detected by measuring temperature changes. As warmer material slides past the sensor, the cooler material that has not yet been heated can be detected.

2.3.3 Robotic Touch Sensors

A review of tactile devices starts with the early work of Inoue and Binford. In 1972, Inoue [54, 55] used an array of switches made using a foam rubber separator and conductive paper. Hill [45] also performed early research with a sensor that used a simple array of switches. Binford [12] pursued several approaches, including semiconductor strain gauges and various pressure sensitive paints and rubber polymers. Since then, a wide variety of technologies have been employed to reduce sensor size, to make them conform to curved mounting surfaces, and to improve their reliability and sensitivity. The technologies that have been tested include:

1. *Resistive:* pressure is detected from a change in a material's resistance (Purbrick [84], Hillis [47], Grotenhuis and Moore [43], Snyder and St. Clair [98], and Bastuscheck [7]).

2. *Capacitive:* pressure is detected from a change in the thickness of a capacitor's dielectric (Boie [13], Siegel et al. [94], Fearing [33], and Jacobsen et al. [56]).

3. *Magnetic:* compression, rotation, and shear are detected from the change in orientation of magnetic dipoles (Hackwood et al. [44], Checinski and Agrawal [19], and Kinoshita [65]).

4. *Optical:* compression is detected from a change in intensity of light (Schneiter and Sheridan [92], and Begej [8]).

5. *Semiconductor:* pressure is transduced using resistive, capacitive, or optical sensor elements integrated onto a chip (Raibert [85], Raibert and Tanner [86], Chun and Wise [22], and Tise [102]).

6. *Polyvinyledene Fluoride:* pressure is detected from an electric response generated when a Polyvinyledene Fluoride material is disturbed (Kinoshita et al. [66], and Dario et al. [28]).

7. *Ultrasonic:* pressure is detected by using an ultrasonic pulse to measure the change in thickness of a material (Grahn and Astle [39]).

8. *Thermal:* material is identified based on its thermal conduction properties (Siegel and Simmons [96], and Russell [88]).

Other interesting recent devices include a slip detector by Howe and Cutkosky [51], and a shear sensor by Novak [78]. As can be seen, the list of tactile sensing technologies is large and varied. Nonetheless, major problems which prevent the use of tactile sensors for all but a few specialized applications still remain.

Today, robotic hands such as those designed by Salisbury [90] and Jacobsen [57] are increasingly being equipped with contact sensors. Jacobsen [56] described a sensing system suitable for covering most surfaces of the Utah-MIT hand with binary contact detectors. Dario et al. [27] has mounted his sensor on a robotic finger. Fearing [34, 35] has

mounted a capacitive-based fingertip sensor on the Salisbury hand. Brock and Chiu [15] also developed a fingertip sensor that has been mounted on the Salisbury hand. Their sensor is the one that is used for the experiments conducted in Chapter 5.

Based on the experiences of tactile sensor developers, it is reasonable to conclude that human-like sensor performance is not yet achievable. The cost of building the sensor systems is also unknown, and is likely to be large.

2. Grasp Planning and Analysis

Grasping and pose determination have an important relationship to each other. Grasping is the acquisition of an object while determination identifies its final resting position. Grasping without determination is incomplete, as the lack of knowledge in how the object is positioned in the hand may preclude subsequent useful manipulations.

Typically, a grasp planner is given as inputs a model of an object, its position, a kinematic model of the robot, and perhaps some task level information. Planners make a number of assumptions to make their task tractable, including simplified models of the fingertip contacts, frictional effects, and the object itself. Only then does planning a grasp become tractable.

Even with simplifications, the planner's task is a hard one. To begin, it must find a set of finger contacts that stably acquire the object. If the object is to be manipulated rather than just constrained, the planner must take this into account. This can be thought of as a task-level constraint. The question of reachability must also be considered. The robot must have a clear path from its starting position to the final grasp position. Collisions between the hand and the grasped object, and the hand and workspace obstacles must be considered. To date, n grasp planners have been able to address all these issues at once.

To make the problem tractable, the grasping task is usually divided into easier sub-problems. Typically, the problem is decomposed into an analysis of stability, feasibility,

and reachability, each which is solved separately. Stability considers whether a particular set of fingertip to face assignments grasp an object stably. Feasibility considers whether the hand's kinematics can achieve the finger positions that the set of contact points requires. Reachability considers whether the robot's arm can position the hand's wrist at the required location. Solving each of these steps separately greatly simplifies matters, though it often leads to generate and test style algorithms. This approach can be wasteful if the workspace is very cluttered, where most stable and feasible grasps will not be reachable.

2.4.1 Stability Analysis

The problem of analyzing and optimizing a given fingertip grasp has been extensively studied. Many criteria have been proposed for a good grasp. Good grasps should be stable. They should resist outside perturbation. A force closure grasp, one where the object is totally constrained by the contacts independent of the magnitude of the contact forces, is often considered ideal (Nguyen [77]).

Many researchers have analyzed grasp stability. Kerr and Roth [63] examined the problem of selecting the internal grasping forces given three contact points. For a three fingered hand grasping an object with just its fingertips, only six of the nine fingertip force unknowns are constrained by the basic Newton-Euler force and torque balance equations. The remaining three force components form the null space of internal grasping forces. Though these forces can be assigned arbitrarily, they suggested various optimization techniques for choosing them, based on a set of constraints. Barber et al. [5] used a quality measure based on the amount of friction necessary to keep the grasp from slipping. Jameson and Leifer [58] predicted the stability of fingertip grasping with both point contact and soft finger contact models. Li and Sastry [68] proposed a task oriented quality measure for evaluating a grasp. Park and Starr [80] proposed two indices for measuring the quality of a grasp. The *uncertainty grasp index* indicates how stability is

effected by position uncertainty. The *task compatibility index* represents how well suited the grasp is for the intended task.

Static stability of a grasp is, of course, not the only issue that must be considered. A good grasp is one that firmly grips the object, and that can resists disturbing forces. This type of grasp, however, might preclude subsequent object motions that are necessary for completing the manipulation. This observation leads to grasp classification schemes such as that proposed by Iberall and Lyons [53]. For example, grasps can be coarsely grouped into *power* grasps and *manipulatory* grasps. A person's normal grip of a hammer can be considered a power grip, while the grip used for holding a pencil is a manipulatory one.

2.4.2 Grasp Synthesis

Unlike much previous work, Nguyen [76] developed an analytical technique for synthesizing force closure grasps, rather than just analyzing given grasps. Essentially, he found regions of contacts for the fingertips that constrain the object according to the force closure criterion.

Jones and Lozano-Pérez [62] studied the problem of grasp selection as a collision avoidance problem. Hence, their work concentrated on the question of reachability. An efficient representation of configuration-space was used. Likewise, Pertin-Troccaz [82] studied this problem using a configuration-space approach. Both groups only examined the case of two fingered grasps.

Pollard [83] developed an entire system to plan a grasp, given an initial approach direction and a desired set of contact faces. Her system efficiently solved the stability, feasibility, and reachability problems using a combination of algorithms and heuristics.

2.4.3 Pre-Shapes and Hand Primitives

With a pre-shape, the basic form of the hand is limited to a small number of predefined configurations. Each pre-shape has a small number of parameters that are used to vary

the configuration. For example, a curl pre-shape with one parameter could be used wrap the hand around an object. In essence, pre-shapes are used to limit the configuration space of the hand. By using pre-shapes, the grasp planning process can be simplified.

A pre-shape provides an initial configuration of the hand that is considered likely to result in a good grasp. This gives a starting point for choosing the fingertip to face assignments. Alternatively, pre-shapes can be used to define an acquisition strategy. The curl pre-shape can be used as a starting hand form. A grasping strategy could simply reduce the *curl* parameter until contact with the object has been made.

An assumption made by many grasp planners is that only fingertip contacts are considered. Certain types of grasps fall, at least partially, into this category. Writing with a pencil is an example, though even here the palm provides important support for the implement. Certainly, a good power grasp would have far more object-hand contacts than just with the fingertips. The fingertip grasp simplification is often made because it makes stability analysis more tractable.

Iberall et al. [52] and Stansfield [100] used knowledge based approaches for grasp planning and pre-shape selection. This type of approach has the potential advantage of being able to incorporate task-level specifications into the knowledge base. Tomovic et al. [103] synthesized grasps by matching the object to a small number of geometric primitives, and selecting a pre-shape based on the primitive.

2.4.4 Acquisition Behaviors

The problem discussed until now has been one of planning and analyzing a grasp of a known object at a known location. An important alternative scenario is one where the object to be grasped is not known, or is known but not at a pre-determined position, or both. This can be called object acquisition, to distinguish it from planned object grasping. We acquire objects as part of our daily repertoire of manipulations. Reaching into our pockets to pull out their contents is an example of such a manipulation.

Chammas [18], for example, addressed this problem. He found the geometric conditions necessary for form closure grasps of cylindrical objects. With his analysis, capture zones can be computed, where in a computed region a particular strategy is guaranteed to grasp the object.

Robots are needed to perform acquisition tasks. NASA is interested in a tool retrieval system that can acquire free floating lost tools. For another space application, NASA is studying reliable systems for grasping beams. In some sense, the role of a parts feeder is to acquire object and to determine their position and orientations. Typically, a feeder is hard to design, and is specific to a particular part. They are built using various tricks, including vibratory bowls. A more general part alignment system might consist of a simple hand-like gripper along with a pose determination system. Not only would this approach give a more flexible system, but a single pose determination algorithm could work for many parts.

Unlike grasp planning, object acquisition has mostly been studied using whole hand grasps. Contacts between the objects and the hand are not simply limited to the finger-tips. The hand is treated as a capture device, where its pre-shape and closing strategy are designed to maximize the chances of capturing the object.

With both planned grasps and acquired grasps, there is always uncertainty as to where the object has finally rested on the fingers. Because of uncertainty, even the best planned grasp will not be executed as expected. By definition, acquired grasps have no knowledge of the object pose. This report address the problem of recovering an object's pose after it has been grasped or acquired.

2.5 Recognition

There are a number of recognition problems that are studied in conjunction with hands. Lower level recognition includes the measurement of local contact properties, such as curvature and texture. Higher level recognition includes object identification and pose

determination. For completeness, this section overviews a variety of the recognition and determination problems, briefly discussing the sensing strategies and the algorithms that have been used to solve them.

The distinction between low level (local) recognition and high level (global) recognition is not only a distinction between the types of problems studied. Rather, it contrasts how the sensor data itself is viewed. One approach treats tactile sensors as miniature vision systems that can give an image of a small object that is being manipulated. Here, tactile sensors are thought best for recovering local object features, such as surface curvature. As an alternative, tactile and kinesthetic sensors can be thought of as providing global information, which is useful for global recognition problems. In both cases, the tactile sensors extract a small piece of information at the contact point. What differs is how the information is used. A local strategy, such as one for curvature detection, might require multiple sensor readings from a small region. The finger would perform a series of motions to obtain this information. A global strategy would combine simultaneous readings from many sensors, and use the information to detect a global property, such as the object's pose.

It is important to note that while low level recognition is almost always studied as the problem of interpreting tactile sensor data, high level recognition can be studied without such data. In particular, this report explores how pose determination can be performed using just kinesthetic sensors, along with the geometric constraints inherent in the problem. Methods which fully exploit kinesthetic sensing are desirable, as they take full advantage of the available data. The addition of tactile sensor data to these methods is certainly desirable, and could be used to improve their reliability and accuracy.

2.5.1 Low Level Recognition

Low level recognition measures local properties at the sensor contact point. A representative set of work in this area is examined here, including the measurement of curvature,

texture, forces, and small features.

Fearing [36] analyzed how a tactile sensor could be used to accurately measure local contact curvature from a set of strain measurements. The problem of combining together a number of contacts to estimate curvature was studied by Brock and Chiu [15] and Montana [72]. Driels [29] found the orientation of a line on a flat contact array. Ellis [30] examined the texture classification problem. He extracted features from a tactile image, much in the same way that a vision system would extract features from an optical image. He created a feature vector which could be used for recognition. Howe and Cutkosky [51] designed and studied a sensor for detecting slip. Bicchi [11] studied the problem of force-based sensing. Russell [88] and Siegel et al. [95] developed thermal sensors that can measure heat conduction properties. This is useful for material identification and slip detection. In one of the earliest active sensing systems, Hillis [47] used his sensor to distinguish a set of nuts and bolts. These parts were all small compared to the size of his sensor. Each part was classified according to three parameters: shape, bumps, and stability. A sensor equipped finger actively probed a part to ascertain its parameters.

2.5.2 High Level Recognition

High level recognition finds global properties of an object. A representative set of work in this area is examined here, including finding surface maps of object, model-based pose determination, and the scheduling of sensor motions.

Kinoshita [64] recognized objects using a hand covered with simple binary sensors, using a pattern classification scheme. Likewise, Okada and Tsuchiya [79] recognized object using patterns from tactile sensors and the joint angles from a hand. Allen [2] built a surface map of an object from multiple local measurements. Gaston and Lozano-Pérez [38] and Grimon and Lozano-Pérez [42] studied model-based pose determination. They assumed sensors that return contact positions and normals, and performed an efficient search of an object pose interpretation tree using constraint propagation methods.

Snyder and St. Clair [98] used a similar approach in their recognition system. Similarly, Schneiter [91] studied an active sensing approach, and developed a method for scheduling sensor motions based on the recognition scheme of Grimson and Lozano-Pérez [42]. Using their notion of an interpretation tree, Schneiter developed a scheme for scheduling sensor moves to remove ambiguities in pose interpretations. Ellis [31] studied the problem of how a robot should proceed when acquired sensory data is insufficient to recognize an object and to determine its pose. Luo and Tsai [69] developed a object recognition system that used a tactile array and a vision system. They found features such as contact moments, and used a decision tree to match object feature vectors.

Tactile recognition of objects, from a theoretical standpoint, can be though of as a geometric probing operation. Skiena [97] studied this problem and extended the earlier work by Cole and Yap [23]. Cole and Yap defined a finger probe to be the first intersection point p between a line l and an object P. The line specifies the path that the finger takes when moving toward the object. In their work, it is assumed that absolute determination of object P is desired. That is, given a model of object P, bounds on the number of probing operations necessary to determine if the unknown object is P are developed. They found that $2n$ finger probes are necessary and sufficient to verify a convex $n-$gon. Intuitively this is true because probing each vertex and edge in the object will determine its position.

Constraint-Based Pose Determination

Chapter 3

3.1 Introduction

This chapter examines a pose determination strategy based on searching an interpretation tree of potential contact assignments. An exhaustive search is avoided by exploiting geometric constraints from the object and the hand, and by knowledge of the grasping strategy. The experiments described in this chapter suggest that an object usually fits into a hand shape only a small number of ways. Adding additional information, such as joint torques, further reduces the number of potential poses, often to just one. The algorithm is on-line, where the computations are done after the hand has completed its grasping operation. By carefully pruning the pose interpretation tree, search time is kept small. This work also suggests that tactile sensors may not be necessary for certain pose determination tasks. Dextrous hands should be able to perform certain useful sensing tasks with just basic joint position information.

This research assumes that the type of grasps being attempted are whole hand grasps. These grasps are the one that humans commonly use, where many surfaces of the fingers

touch the object. Even in cases of fine motion manipulation, fingertip grasps alone are uncommon. As an example, take writing with a pencil. While motion of the fingertips that are in contact with the implement is a crucial part of the manipulation, other surfaces of the hand provide necessary support. Most hand control research to date has used fingertip grasps, where only fingertip surfaces touch the grasped object. As will be seen, whole hand grasps have far more information content than fingertip grasps. They provide a large number of geometric constraints that can be exploited by object recognition systems.

3.1.1 Relevance of this Problem

Pose determination is helpful for verifying that a planned grasp has been executed correctly. An exploratory robot might use small set recognition and pose determination to gather information about its environment. The constraint-based pose determination method examined in this chapter helps understand how much additional sensor data is really required for this type of problem.

3.1.2 Why Use Geometric Constraints?

The recognition method studied in this chapter relies heavily on the geometric constraints that are obtained from models of the hand and objects. There are several reasons why so much emphasis is placed on these geometric constraints:

1. They are essentially free, since the kinematics of the hand and models of the objects are known.

2. They are powerful and will greatly reduce the space of possible poses (see Grimson [41]).

3. They play a complementary role to sensor data, helping to confirm possible interpretations and resolve ambiguities, which results in a more robust system.

The shape of the hand provides many global clues as to how an object might be oriented. Imagine grasping an object that is much larger in one dimension than in the other. The separation distance between the fingers and an opposing thumb will probably rule out certain orientations of the object. For example, if two particular finger links are involved with a grasp, a simple distance constraint will determine which parts of the object can be placed between them.

3.1.3 Why Use Just Joint Angle and Torque Data?

Many potential haptic data sources can be used for input to a localization algorithm, including joint angles, joint torques, wrist forces, tactile data and visual data. When deciding if a data source should be employed in a recognition algorithm, one must consider the cost and difficulty of obtaining the data, among other factors. Importantly, minimal sensing recognition approaches allow a better understanding of the problem's inherent constraints. A strategy that works well with limited data will only benefit from additional information if it is available.

The methods described in this chapter utilize just joint angle and joint torque data. It is important to explore the full power of these particular data sources since they are so readily available. Almost all robots provide joint angle and joint torque sensors as part of their normal control system. A primary goal of this chapter is to investigate how much haptic information content is present in this data, since it is available essentially for free.

3.1.4 Chapter Overview

The following sections show how constraints are used for object pose determination. Section 3.2 provides an overview of previous work. Section 3.3 outlines the assumptions required for the solution given. The approach is discussed in Section 3.4. Section 3.5 presents a number of simulations of the algorithm. Section 3.6 discusses the experimen-

tal setup and the results. Potential extensions of the approach for problems in three dimensions are presented in Section 3.7. Section 3.8 presents conclusions.

3.2 Related Work

The recognition strategies used in this chapter are most closely based on the work of Gaston and Lozano-Pérez [38] and Grimson and Lozano-Pérez [42]. They decompose recognition into an efficient search of an object pose interpretation tree using constraint propagation methods. A set of feasible object poses are generated, and then tested for validity using an additional set of verification constraints. While the approach described in this chapter uses a similar notion of an interpretation tree, it uses a different set of data sources. Their method relies on knowledge of contact locations and normals. This work assumes a much weaker set of sensor inputs. Importantly, no explicit contact sensor information is used.

3.3 Assumptions

The following assumptions are made in this chapter:

- The hand has been modeled.
- The objects have been modeled using polyhedra.
- The grasped object is assumed to be in static equilibrium.
- Hand joint angle sensor data is available.

3.4 Approach

This section describes the method used for determining the pose of a grasped object. The algorithm uses a generate and test paradigm, where candidate poses are hypothesized and then tested for validity. Thus, this description is broken into two components, the generator and the tester.

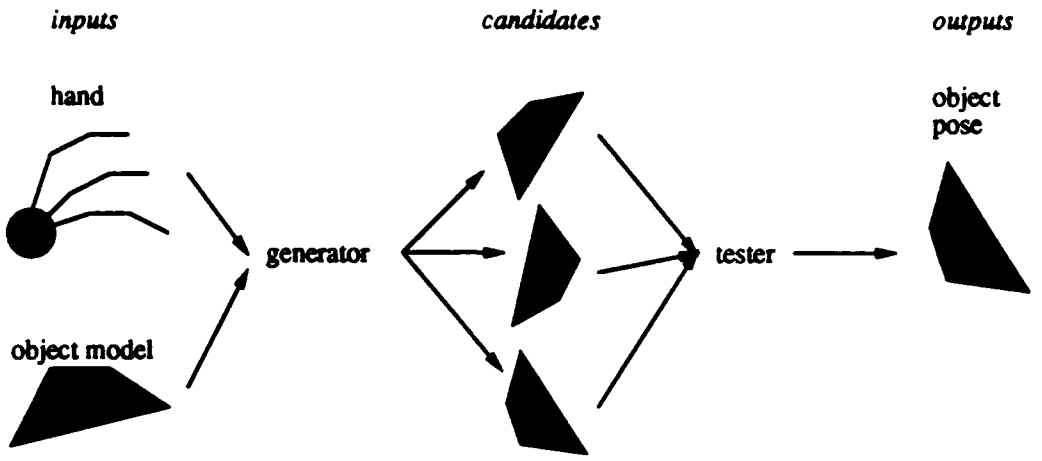


Figure 3.1: *Overview of the constraint-based recognition method. Pose candidates are generated from the hand shape. A tester removes the candidates that are inconsistent.*

The algorithm described is for planar hands and planar, polyhedral objects. As will be seen, the algorithm can also be used for three-dimensional polyhedral objects that are resting on a table on one of their faces. A full three-dimensional extension of this method should also be possible, and is discussed in Section 3.7.

For a generate and test method to be useful, the generator must come up with a reasonably small set of candidates for testing. The generator must perform its job efficiently, as there is a large space of possible solutions for it to traverse. A useful generator must also be complete, in that it should never prune the correct solution from the interpretation space (Grimson and Lozano-Pérez [42]). The most important criterion for the tester is for it to incorporate all additional available constraints into its tests. It need not be particularly efficient, since it should have relatively few candidates to evaluate.

A brief overview of the algorithm is presented here, giving a road map for the remainder of this section. The algorithm is present as two distinct modules, the generator and the tester:

1. Pose Generation:

   (a) Find consistent object vertex pairs that can be placed on finger link pairs, using a distance constraint.

   (b) Find consistent object vertex triplets that can be placed on finger links, using the vertex pairs.

   (c) Determine the orientation of the object vertex triplet triangle. It will be shown that each vertex triangle can have only a finite number of placements on the finger links.

   (d) Find the orientation of the entire object, based on the orientation of the vertex triplet triangle.

2. Pose Testing:

   (a) Verify that the generated object pose and hand are free of intersections.

   (b) Verify that the generated object pose is consistent with the joint torque sensor data.

Figure 3.1 diagrams the method, and shows the flow of information, from inputs to outputs.

3.4.1 Constraining an Object's Position

The problem of geometrically fixing the position of an object in a hand can be decomposed into the problem of assigning object vertices to finger segments. There are two types of assignments that must be considered (see Figure 3.2):

1. Three object vertices can be placed on three finger segments (Figure 3.2 A).

2. Two object vertices can be placed on one finger segment, and another vertex placed on a second segment (Figure 3.2 B).

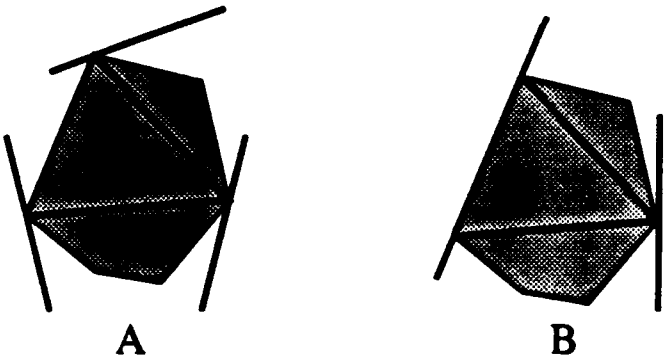


Figure 3.2: *Two types of grasping constraints. Object A is grasped with three vertices on three finger segments. Object B is grasped with two vertices on one finger segment and another vertex on a second segment. The object vertex triangle is shown for each grasp.*

In both these cases, the position of the object vertex triangle, as drawn in Figure 3.2, is fixed with respect to the fingers. That is, specifying either of the two classes of contacts fixes the position of the object. Note that this method does not just place an object that is triangular in shape. Rather, the triangle formed by any three of an object's vertices is placed, which constrains the position of the object as a whole. This placement cannot be made when any two of the finger segments are parallel, as the vertex triangle is not fully constrained.

The object initially has three degrees of freedom, one rotational and two translational. By placing an object vertex on a finger edge, one degree is constrained. Thus, assuming vertex contacts, at least three vertices must be placed to fully constrain the object. Placing an edge of the object on an edge of the finger results in two degrees of constraint. Thus, placing one edge and one vertex, or two edges, fully constrains the object's position to a discrete set, unless the case is degenerate (e.g. parallel edges).

Note that the constraints discussed are different from grasping constraints. That is, just because an object's position is specified by its finger contacts, it may or may not be stably grasped. To determine if a grasp is possible, one must consider other factors, including the type of contact. In the case of a point contact, one must consider

if it is with or without friction. For the purposes of this recognition strategy, these considerations are ignored. Instead, the geometric constraints necessary to specify an object's location are considered, not the contact conditions necessary for the grasp to be stable.

3.4.2 Pose Generation

The pose generation process is based on an interpretation tree, as developed by Gaston and Lozano-Pérez [38]. The interpretation tree represents all possible assignments of object vertices to finger edges. Nodes in the tree represent fingers segments and links represent object vertices. Thus, a node and a link together represent an object vertex to finger segment assignment. The depth of the tree is equal to the number of object vertices. Each node has one child for each finger link segment, plus a special *no-contact* node, as used by Grimson and Lozano-Pérez [42]. A fully expanded interpretation tree represents all possible assignments of object vertices to finger edges, without regard to the feasibility of the assignments. Only certain branches of the tree correspond to feasible assignments. The trick to using an interpretation tree efficiently is to generate only its feasible nodes. The gist of this method is how to selectively perform this expansion.

An object grasped by a hand, and the corresponding interpretation tree, is shown in Figure 3.3. The hand has three links, and the object has three vertices. Only the feasible portions of the tree are diagrammed, where feasible means assignments that satisfy a particular set of constraints. Branches representing placements that are not feasible have been omitted from the drawing. Later in this section, the methods used for determining the branches that should be pruned are discussed. For now, just assume that such pruning methods exists.

The actual contact assignment between the fingers and the object, as diagrammed in Figure 3.3, corresponds to the left most branch in the interpretation tree. This path is highlighted. Reading from the root to the final leaf, the contacts are $V_1 \rightarrow F_1$, $V_2 \rightarrow F_3$,

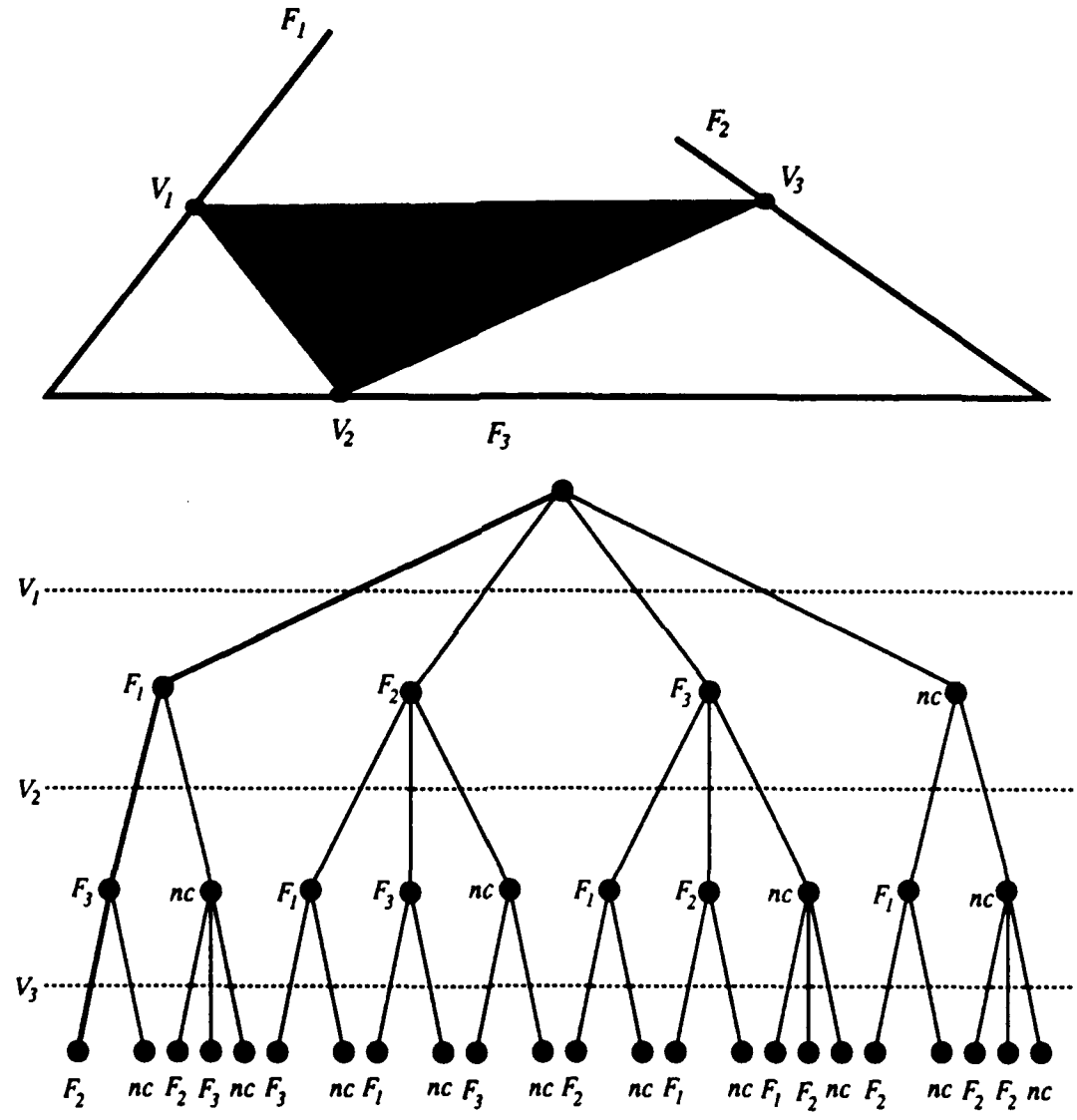


Figure 3.3: *A pose interpretation tree for a grasped object. The object is composed of three vertices. The hand has three finger segments (two fingers and a palm). A branch from a parent node (object vertex) labeled with a finger indicates a placement of that vertex on that finger. This tree only contains assignments of one vertex to each finger segment.*

and $V_3 \rightarrow F_2$. Other branches in tree represent other feasible contact assignments. For example, the second to left most branch represents the contact assignment $V_1 \rightarrow F_2$, and $V_2 \rightarrow F_3$. In this case, V_3 is not in contact with any of the finger links.

In general, a connection from a link to a node indicates an assignment of an object vertex (the link) to a finger segment (the node). A connection from a node to a no-contact node indicates that the object vertex is not in contact with that finger segment. A path from the root of a tree to any intermediate non-terminal node represents a partial assignment of an object's vertices to finger segments. A path from the root of a tree to a leaf indicates a full assignment of all object vertices to finger segments. Thus, any node in the tree represents a potential contact assignment. The path from the root to that node specifies the particular assignment.

Any node that has at least three vertex-finger assignments constrains the position of the object, in a geometrical sense. The role of the pose generator is to efficiently build this interpretation tree and find such candidates, and to pass them to the verifier. The next sections discuss the constraints that are used to prune inconsistent paths from this tree.

Finding Consistent Vertex Pairs

The first of the two tree pruning constraints used is called the *vertex pair* constraint. A pair of object vertices can be placed on a pair of finger edges when there is at least one place where the length of a line drawn between each finger is equal to the distance between the vertices. Figure 3.4 shows where such a line can be drawn between two particular edges. All vertex-finger assignment pairs in a path in an interpretation tree must satisfy this criterion. If an assignment fails this test, the path in the tree is pruned. An efficient way to compute the ranges of position on each finger where such an fixed-length line can be drawn is now presented.

Let r_1 be the ray of finger edge F_1 and r_2 by the ray of finger edge F_2, as shown in

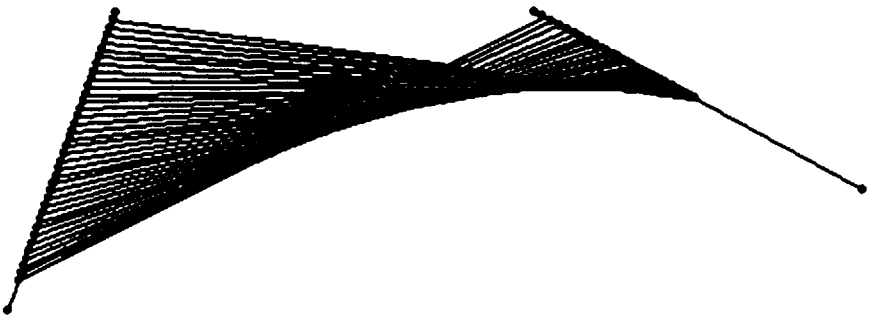


Figure 3.4: *Possible placements of a vertex pair on finger edges. This diagram shows the family of positions where a pair of object vertices (shown as a fixed length line) can be positioned between two finger edges.*

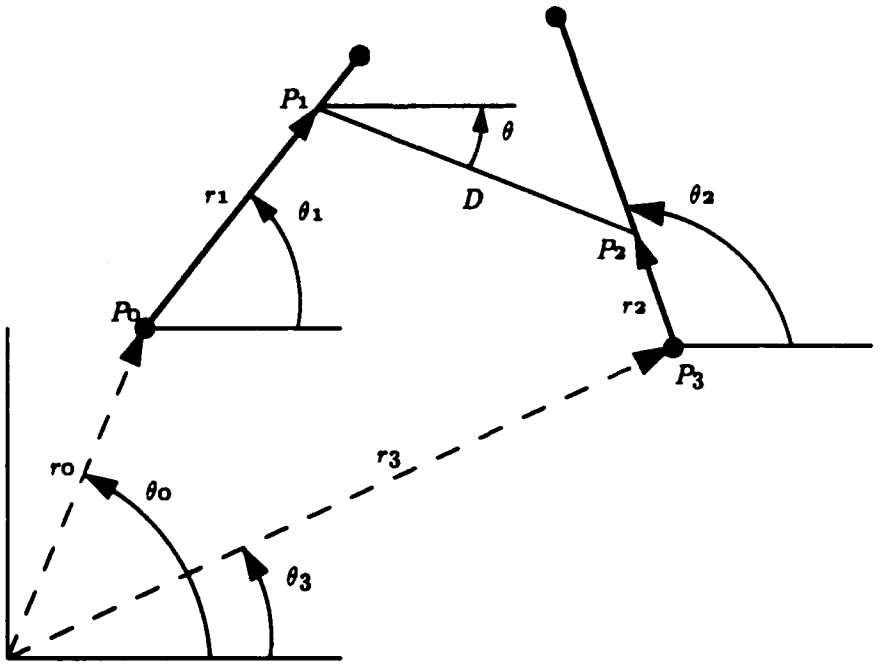


Figure 3.5: *Distance constraint coordinate system. A complex coordinate scheme, shown here, is used to compute the distance constraint.*

Figure 3.5. The range of values of r_1 such that a line of fixed length D can be drawn from r_1 to r_2 is desired. To begin, note the equations for the endpoints of the edges and the fixed length line:

$$P_0 = r_0 e^{i\theta_0} \tag{3.1}$$

$$P_1 = P_0 + r_1 e^{i\theta_1} \tag{3.2}$$

$$P_2 = P_1 + D e^{i\theta} \tag{3.3}$$

$$P_3 = r_3 e^{i\theta_3}. \tag{3.4}$$

P_2 can also be written as

$$P_2 = P_3 + r_2 e^{i\theta_2}. \tag{3.5}$$

By substitution using the above equations, r_2 can be obtained in terms of r_1

$$r_2 = r_0 e^{i(\theta_0 - \theta_2)} + r_1 e^{i(\theta_1 - \theta_2)} + D e^{i(\theta - \theta_2)} - r_3 e^{i(\theta_3 - \theta_2)}. \tag{3.6}$$

Expanding, and setting the complex part of the solution to 0 gives

$$0 = r_0 \sin(\theta_0 - \theta_2) + r_1 \sin(\theta_1 - \theta_2) + D \sin(\theta - \theta_2) - r_3 \sin(\theta_3 - \theta_2). \tag{3.7}$$

Solving for θ,

$$\theta = \sin^{-1}\left(\frac{r_3 \sin(\theta_3 - \theta_2) - r_1 \sin(\theta_1 - \theta_2) - r_0 \sin(\theta_0 - \theta_2)}{D}\right) + \theta_2 \tag{3.8}$$

and noting that $\sin^{-1} x$ is defined only for $-1 \leq x \leq 1$ we obtain:

$$-1 \leq \frac{r_3 \sin(\theta_3 - \theta_2) - r_1 \sin(\theta_1 - \theta_2) - r_0 \sin(\theta_0 - \theta_2)}{D} \leq 1. \tag{3.9}$$

Thus we conclude that r_1 can take on values in the range:

$$\frac{D - K}{\sin(\theta_1 - \theta_2)} \geq r_1 \geq \frac{-D - K}{\sin(\theta_1 - \theta_2)} \tag{3.10}$$

where,

$$K = r_0 \sin(\theta_0 - \theta_2) - r_3 \sin(\theta_3 - \theta_2). \tag{3.11}$$

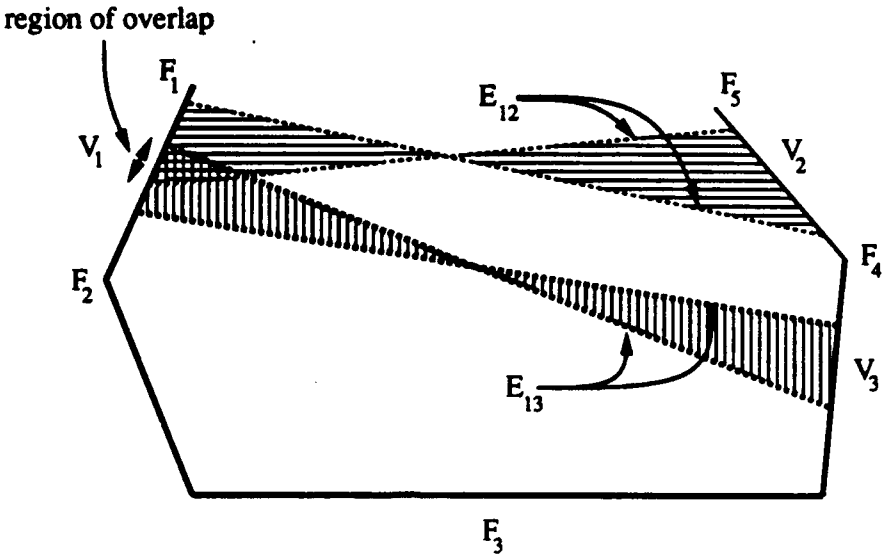


Figure 3.6: *Consistent vertex pairs. Three object vertices placed on three finger edge segments are shown. Vertex V_1 from both pairs shares a common edge, and overlap.*

Equation 3.10 gives the range of positions along finger edge F_1 where a line of length D can be drawn to finger edge F_2. For convenience, this equation can be represented by a function R:

$$R(F_1, F_2, D) = F_1', \tag{3.12}$$

where F_1' is the portion of F_1 where the line of length D can be placed. If $\|F_1'\| = 0$, then the placement is not possible. Thus, when given a pair of vertex-finger assignments, Equation 3.12 provides a fast check for distance consistency. From the viewpoint of the interpretation tree, this test validates the consistency of a link.

Finding Consistent Vertex Triplets

Two pairs of vertex-finger assignments are considered consistent if one of the vertex-finger assignments from each pair is the same, and if they overlap in placement on the common finger. This can occur between two adjacent links in a path in an interpretation

tree. The diagram of a hand and potential object vertex placements shown in Figure 3.6 helps explain this.

Two vertex pairs are shown, $\{V_1, V_2\}$ and $\{V_1, V_3\}$, where V_1 is on finger F_1, V_2 is on finger F_5 and V_3 is on finger F_4. The lines E_{12} and E_{13} are the equal length distances between each pair of object vertices, where $E_{12} = \|\overline{V_1V_2}\|$ and $E_{13} = \|\overline{V_1V_3}\|$. In this case, the vertex triplet condition is met because for both pairs, vertex V_1 is on finger F_1, and there is an overlap of the placement range of the shared vertex, V_1.

More formally, the overlap range can be computed using the vertex-pair constraint, as defined in Equation 3.12. First, the placement range for each pair of edges is computed:

$$F'_{1a} = R(F_1, F_5, E_{12}) \tag{3.13}$$

$$F'_{1b} = R(F_1, F_4, E_{13}), \tag{3.14}$$

where F'_{1a} and F'_{1b} are the subranges of F_1 where each placement can be made. The overlap range can be found be computing their intersection,

$$F'_1 = F'_{1a} \cap F'_{1b}, \tag{3.15}$$

where F'_1 is the desired part of edge F_1 where the common placement can be made.

Finally, the subranges of F_5 and F_4 that are compatible with this placement are easily computed using function R:

$$F'_5 = R(F'_1, F_5, E_{12}) \tag{3.16}$$

$$F'_4 = R(F'_1, F_4, E_{13}). \tag{3.17}$$

Thus, this test, when given two vertex-finger pairs that share a common placement, provides a fast consistency check. While the previous constraint tested the consistency of a single link in an interpretation tree, this test is used to verify the consistency of a pair of links. The next section will describe in more detail how the vertex pair and vertex triplet constraints can be used to efficiently build an interpretation tree.

Efficiently Building the Interpretation Tree

At this point we describe how to efficiently build an interpretation tree using the vertex pair and vertex triplet constraints. To build the tree, each node is expanded, starting from the root, in a breadth first search. A node will have one child for each of the finger segments in the hand, along with an additional child for the no-contact case. Each child will be tested for consistency using the constraints developed in the previous two sections, and is pruned if it does not pass the tests.

The vertex pair represented by the link from the new node to its parent is considered. Pair elements on different finger segments are tested using the vertex pair constraint. If the pair elements are on the same segment, the distance between the vertices must simply be less than or equal to the length of the segment. If the pair elements are consistent, the placement ranges are noted, and the link is added to the tree. If the new node is inconsistent, the tree is pruned at that node. The vertex triplet test is then applied to any groups of three or more vertex-finger assignments in the path. This process finds the new subranges of each finger segment that are consistent with the placements on the path. If any of the subranges are null, the new node is pruned. If the tests are all successful, any three vertex-finger assignments on the path can be used to compute an object pose.

To better understand how this process works, consider a path from an interpretation tree that contains three assignments: $F_1 \rightarrow V_1$, $F_2 \rightarrow V_2$, and $F_3 \rightarrow V_3$. The next few paragraphs will detail how these assignments are added to the tree.

The first assignment of $F_1 \rightarrow V_1$ is arbitrarily made. Assume that a new leaf is being evaluated that assigns $F_2 \rightarrow V_2$. Thus, the path from that leaf to the root assigns $F_1 \rightarrow V_1$ and $F_2 \rightarrow V_2$. To test the new assignment, the placement subrange of F_2 is computed:

$$F_2' = R(F_1, F_2, \|\overline{V_1 V_2}\|). \tag{3.18}$$

If $\|F_2'\| > 0$, the node is added to the tree. The subrange F_2' is stored at the node for

future use.

Assuming that the assignment of $F_2 \rightarrow V_2$ was validated, let us evaluate a new assignment of $F_3 \rightarrow V_3$. First, a vertex pair test between the new leaf and its parent is performed, using the parent's current subrange:

$$F_3' = R(F_2', F_3, \|\overline{V_2V_3}\|). \tag{3.19}$$

If this test passes, the vertex triple constraint can be applied between the three edges in the path, where the assignment of $F_2 \rightarrow V_2$ is common between them. Essentially, the range intersection on F_2 between the assignment of $F_1 \rightarrow V_1$ and $F_3 \rightarrow V_3$ is computed. If the length of this intersection is greater than zero, the triple is accepted. This process continues until all nodes have been expanded or pruned.

By using these pruning tests, only a small portion of the full interpretation tree is usually generated. For objects that lack symmetry most vertex-finger assignments are *not consistent* with the tests. It is important to note that this pruning operation will never remove a valid solution that has the types of contacts that are being considered. An actual object placement must be consistent with the tests, and will be found in this generation stage.

Computing the Orientation of the Object

The tree generation from the previous section found two types of vertex-segment assignments. In the first case, three vertices were placed on three different edges. In the second case, two vertices were placed on one edge, and the third was placed on a different edge. This section explains how the pose of the object is recovered from these assignments.

First, consider the case of triplets of vertex to finger segment assignments. The triangle formed by each vertex triplet is fully constrained by the finger edge segments. Thus, the position and orientation of the triangle, and hence the object, can be directly computed. As will be shown, there are potentially four solution classes, and each can have two solutions.

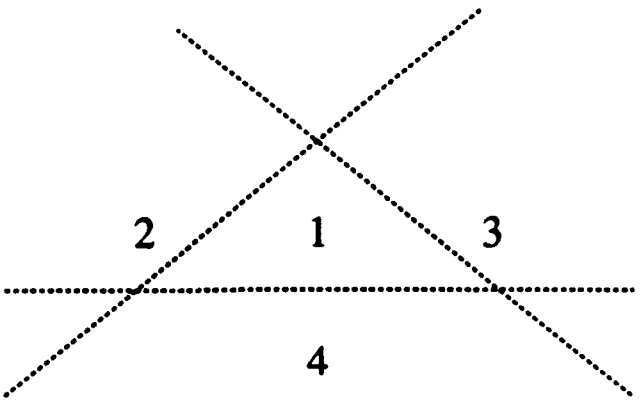


Figure 3.7: *Solution classes for vertex triplets placed on finger segments. The finger segments are extended into the dotted lines. Each of the labeled classes can potentially contain a placement for the object vertex triangle.*

Figure 3.7 diagrams the solution classes. The three finger edge segments being considered are extended to form the dotted lines. The four classes that are created by the intersection of the lines are labeled. The goal of this section is to determine the orientation of a triangle where each of its vertices are constrained to be on one of the three lines. Different solutions for the triangle orientation can be found for each of the labeled classes. Since the triangle is actually being placed on the finger edge segments, and not on an infinite line, solutions need be computed only for the classes where the actual finger segments are present on all three of its boundaries. In addition, after a potential solution has been found, the triangle vertices must be tested to insure that they fall on portions of the finger edges.

An example of a grasped objects and the corresponding solution classes for three different sets of triplets of links is shown in Figure 3.8. The object is shown at the top of the figure. Three possible sets of constraint edges and the corresponding solution classes are shown below the object. In the first set, the three finger segments that are extended with dotted lines are being considered. The extended lines form the four solution classes. Class one, three, and four do not have any portion of the actual finger

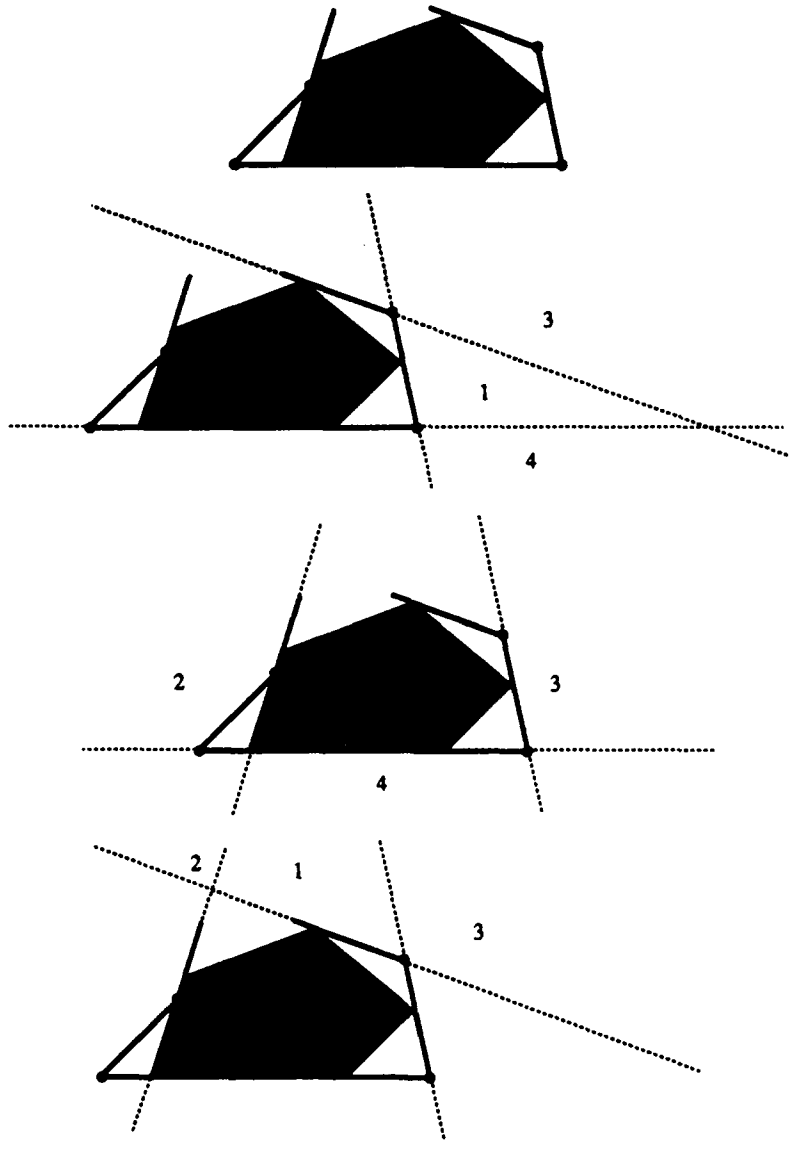


Figure 3.8: *An object vertex triangle and its potential solution classes. The upper figure shows a sample object gripped by a two fingered hand. The lower figures shows the triangle formed by three of the object vertices and the classes formed by the finger edges.*

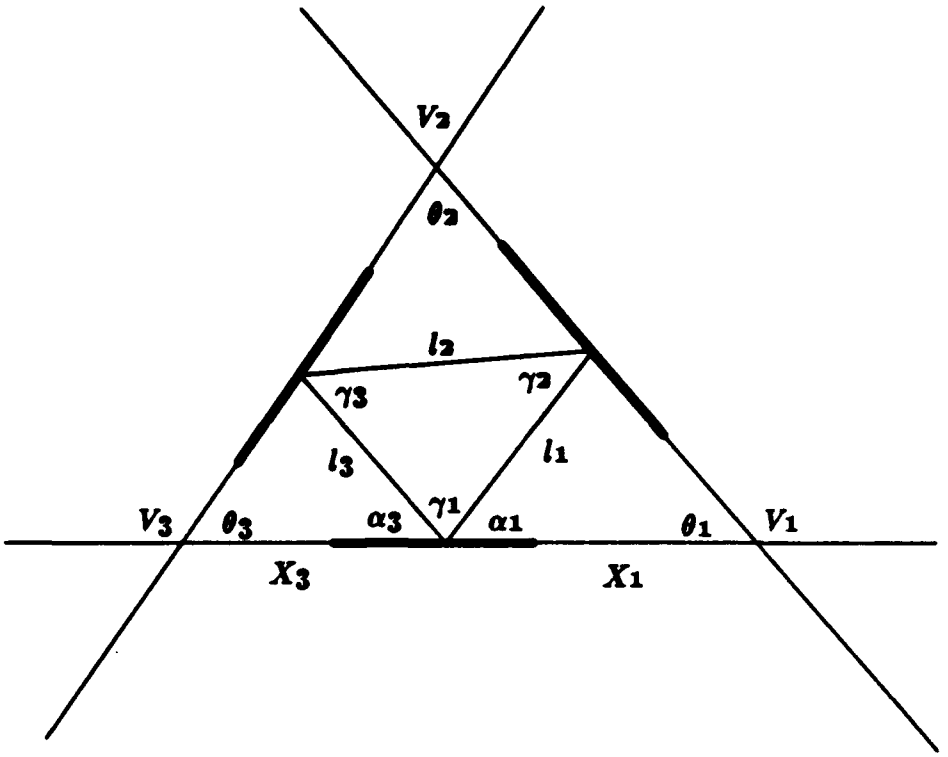


Figure 3.9: *The notation used for computing the orientation of a three-edged triangle. The inner triangle is formed between the three object vertices. The outer lines are formed by the finger edges. The highlighted portion of the lines indicate the location of the finger segments.*

segments extending into them, so the object cannot be positioned in them. Class two has portions of the finger segments extending onto all three of the lines, and thus can potentially contain the object (and in fact, it does).

For clarity, a brief recap may be helpful. At this point a potential assignment of object vertices to finger edge segments has been hypothesized. The pose of the object that would result from this assignment is what is being computed. There are two possible object poses for each of the four potential solution classes. A solution class must be considered if part of each finger segment extends into the edges that bound the class. What remains to be shown is the actual computation necessary to recover the object's pose in a particular solution class.

First, consider the case where three object vertices are placed on three finger edges. Assume that the three finger edges are joined as shown in Figure 3.9. The orientation of the object triangle enclosed by the edges must be found. Note that while the notation used is defined for solution class one (see Figure 3.7), the analysis is the same for the other three solution classes. As shown in Figure 3.9, θ_1, θ_3, and $\overline{V_1V_3}$ are givens. A solution for α_1 and x_1 is desired. By inspection we obtain

$$\tan\theta_1 = \frac{l_1 \sin\alpha_1}{x_1 - l_1\cos\alpha_1} \tag{3.20}$$

$$\tan\theta_3 = \frac{l_3 \sin\alpha_3}{x_3 - l_3\cos\alpha_3}, \tag{3.21}$$

where x_1, x_3, α_1, and α_3 are unknown. In addition, we note that

$$\alpha_1 + \alpha_3 + \gamma_1 = \pi \tag{3.22}$$

$$x_1 + x_3 = \overline{V_1V_3}, \tag{3.23}$$

where γ_1, and $\overline{V_1V_3}$ are known. Solutions for α_1 and x_1 can now be found. Solving Equations 3.20 and 3.21, we obtain

$$A\cos\alpha_1 + B\sin\alpha_1 = C \tag{3.24}$$

where,

$$A = l_1\tan\theta_3 + l_3\cos(\pi - \gamma_1)\tan\theta_3 + l_3\sin(\pi - \gamma_1) \tag{3.25}$$

$$B = \frac{l_1\tan\theta_3}{\tan\theta_1} + l_3\tan\theta_3\sin(\pi - \gamma_1) - l_3\cos(\pi - \gamma_1) \tag{3.26}$$

$$C = (x_1 + x_3)\tan\theta_3. \tag{3.27}$$

Let

$$\phi = \arctan(B, A) \tag{3.28}$$

$$r = \sqrt{A^2 + B^2}. \tag{3.29}$$

Using the relationships

$$\tan^{-1}a = \sin^{-1}\frac{a}{\sqrt{1+a^2}} = \cos^{-1}\frac{1}{\sqrt{1+a^2}} \tag{3.30}$$

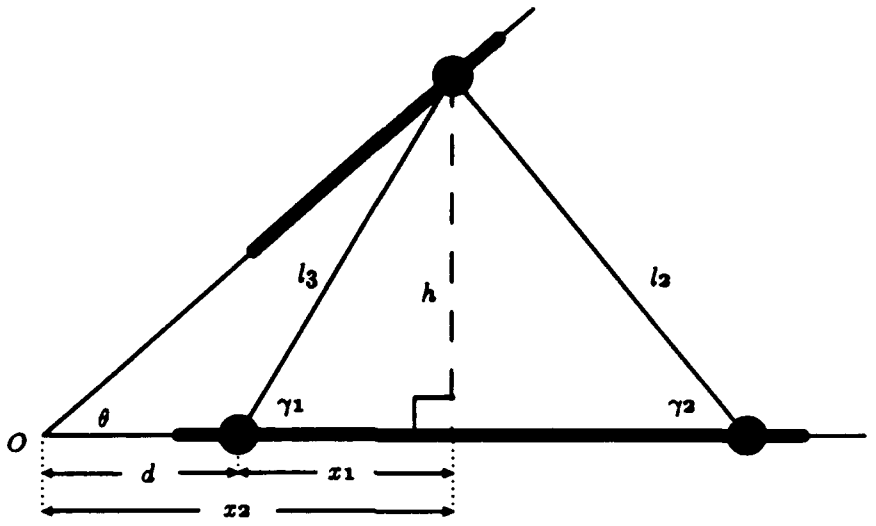


Figure 3.10: *The notion used for computing the orientation of a two-edged triangle. The two finger edges are joined at vertex O. The object triangle is placed between them as shown.*

we note that Equation 3.24 can be written as

$$r\cos(\alpha_1 - \phi) = C, \tag{3.31}$$

which gives

$$\alpha_1 = \cos^{-1}\left(\frac{C}{r}\right) + \phi. \tag{3.32}$$

Finally, x_1 can be obtained from Equations 3.20 and 3.32:

$$x_1 = l_1\left(\frac{\sin\gamma_1}{\tan\theta_1} + \cos\gamma_1\right). \tag{3.33}$$

Next, consider the case of two vertices assigned to one edge, and the third assigned to a different edge. Again, the position of the triangle formed by the three object vertices is fully constrained by the two finger edges. Finding the object position is much simpler in this case because the line formed by connecting two of the vertices is known to fall on one of the finger segments.

If one object vertex is assigned to one finger edge, and an object edge to another finger edge, the orientation of the resulting triangle is easily obtained. Assume that the

two finger edges are extended to lines that intersect at point O, as shown in Figure 3.10. The position of the triangle can be computed from the following equation:

$$d = x_2 - x_1 = l_3 \left(\frac{\sin \gamma_1}{\tan \theta} - \cos \gamma_1 \right). \tag{3.34}$$

3.4.3 Pose Testing

The previous section described how to generate possible pose candidates. This section describes how to verify that the postulated candidates are reasonable. Verification is necessary because the constraint-based procedure for generating poses does not utilize all information available. Rather, the generator uses the constraints that are both computationally inexpensive and that have adequate pruning power. The verifier uses the additional information available from geometric and from grasp acquisition strategy *constraints for further pruning* of the candidates.

Any candidate poses that pass the verification tests are accepted. All others are rejected. Hopefully, only the candidate that corresponds to the object's true position will remain. Experiments conducted in subsequent sections of this chapter will show how well this method performs. The next two sections describe the verification tests in more detail.

Geometric Intersection Test

The object candidates can potentially intersect the hand, as shown in Figure 3.11. The generator simply places triangles on finger segments. From the triangle, the placement of the entire object is computed. Nothing prevents parts of the object from intersecting the hand. The pose tester computes the intersection of the object and the hand, and will reject the candidate if the space is not null. From an implementation standpoint, a threshold is used to select the intersection tolerance that is acceptable.

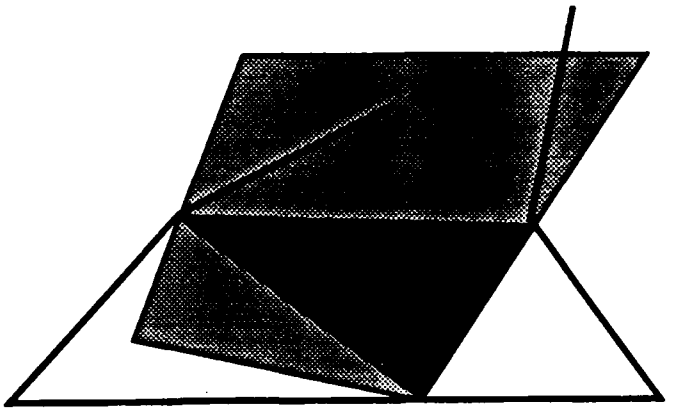


Figure 3.11: *Geometric verification constraint. The object triangle was placed by the algorithm, resulting in the object pose shown. Since the object intersects the hand, it is rejected.*

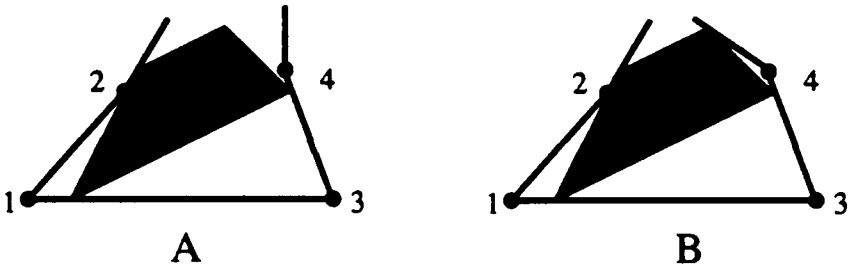


Figure 3.12: *Joint torque verification constraint. Grasp A cannot occur because all joints were programmed to move to a torque limit. Joint 4 would not be at a limit for this grasp. Rather, the hand shape that would result is shown in grasp B.*

Grasp Acquisition Strategy Test

A grasp acquisition strategy can be selected that provides additional information for pose determination. For example, the move-until-contact strategy curls each joint forward, until motion of the joint is no longer possible. A joint torque sensor on the robot may be required for implementing such a strategy. As an alternative, the strategy can be designed into the mechanics of the robot, as Greiner [40] has done with her device.

The diagram shown in Figure 3.12 is used to help understand how this strategy

can verify pose candidates. The move-until-contact strategy provides a guarantee that all joints are constrained from further motion. In this figure, grasp A violates this constraint, as joint 4 is capable of motion. The distal link is free to move until it has collided with the object as shown in grasp B. Thus, if the pose shown in grasp A was generated, and if a move-until-contact grasping strategy was used, the pose would be rejected.

In general, the grasp acquisition strategy test is implemented by simulating the actual grasping strategy. For each candidate pose generated, the grasping strategy is simulated. If the resulting hand shape matches the actual hand shape, the pose is accepted. Otherwise, it is rejected. It is important to note that to use this constraint successfully, a good grasp simulator must exist. If the simulator produces erroneous results, the test could reject the correct pose, or accept incorrect poses.

In general it is hard to create a good grasp simulator. For certain strategies, including move-until-contact, the simulation process is easier. To further reduce the chances of incorrect simulations, the following procedure was used. Rather than to perform a full simulation of a move-until-contact grasp, the resulting grasp and pose were simply analyzed. Each joint, from distal to proximal, was moved forward a small amount. The joint's link was then tested for collision with the object. If a collision did not occur, the pose was considered to be in invalid. The process was repeated for all joints.

3.5 Simulations

This section examines the performance of the pose determination algorithm on simulated grasps. The grasp simulator uses a move until contact grasping strategy. The joints on a finger, from proximal to distal, are moved forward, until the joint's link makes contact with the object, or until it reaches a limit. The number of joints, and their length, can be varied. The simulator supports single or double jointed fingers. For these runs, double jointed fingers were used. Actual robotic hands come in both flavors. The Salisbury

hand, for example, is double jointed. The Utah-MIT hand is not.

By using simulations, a large number of trials can be performed in a systematic manner. The problems caused by poor kinematic models of the robot are also avoided, which is helpful for initial testing. Though the simulations are useful, they cannot substitute for experiments using actual hardware. Such experiments are described in the next chapter.

For each of the simulations, a table summarizing the computations performed is presented. The terms used in the table are described here:

- *Vertices* are the number of vertices in the object.
- *Finger segments* are the number of finger segments (links), including the palm.
- *Expanded nodes* are the number of nodes in the interpretation tree that were examined.
- *Expanded paths* are the number of paths in the interpretation tree that were examined.
- *Placement paths* are the number of paths in the tree that were generated.
- *Full tree paths* are the number of paths in a fully expanded interpretation tree.
- *Generated poses* are the number of poses generated, both those verified and those rejected.
- *Verified poses* are the number of poses that passed both verification tests.

Simulation 1: For this simulation, a two-jointed hand was used. This is the minimal case, where only two numbers are being provided to the pose generation module. The object has five vertices, and is small enough to be enclosed by the hand. Table 3.1 summarizes the simulation results. Due to the rather small size of the problem, a significant percentage of the interpretation tree's paths were expanded. Only one of the generated poses was verified, as shown in Figure 3.13. This pose corresponds to the

vertices	5
finger segments	3
expanded nodes	833
expanded paths	584
placement paths	258
full tree paths	1,365
generated poses	16
verified poses	1

Table 3.1: *Simulation 1: Summary of the interpretation tree.*

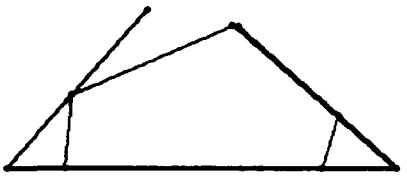


Figure 3.13: *Simulation 1: generated and verified pose. This pose corresponds to the correct grasp of the object.*

object's grasped position. The entire set of generated and rejected poses are shown in Figure 3.14. All these poses were rejected because they are have geometrical inconsistencies. Note that the middle pose on the second row is a fairly good fit, and with a larger "slop" factor, would have been accepted.

Simulation 2: As with the previous simulation, a two-jointed hand was used for this trial. In this case, a more complex object, with eight vertices, was grasped. The object is larger in size than the object from previous trial. Table 3.2 summarizes the simulation results. As the table indicates, the larger number of vertices results in an increase in the potential vertex placements, though the consistent placements remain rather small.

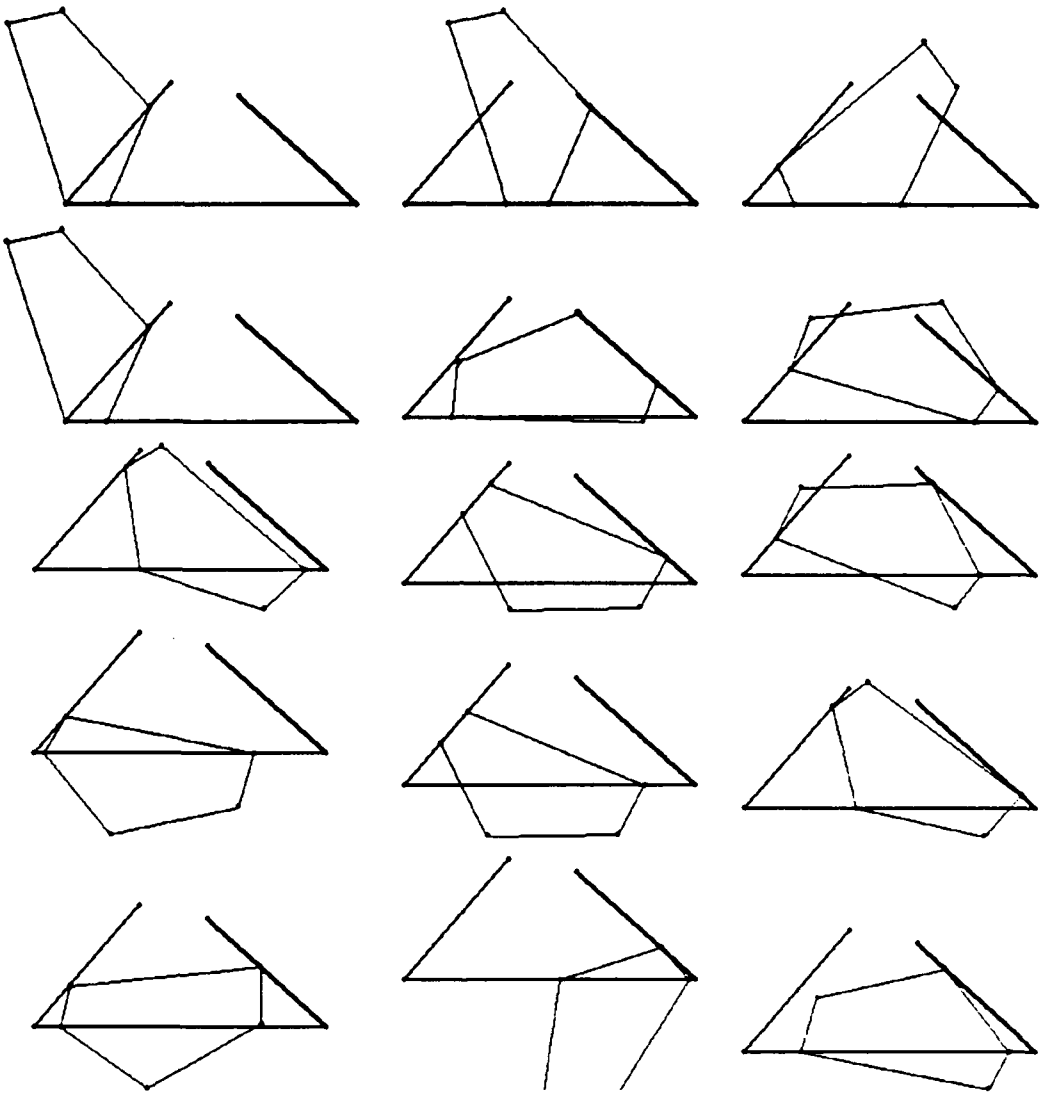


Figure 3.14: *Simulation 1: generated and rejected poses. The entire set of generated and rejected poses are shown here.*

Three of the generated poses were verified, two of which are shown in Figure 3.15. The first pose corresponds to the object's actual position. A small selection from the 105 generated and rejected poses are shown in Figure 3.16. The first pose, in the upper-left corner, was accepted by the geometric consistency check, but rejected by the joint torque constraint check.

vertices	8
finger segments	3
expanded nodes	17,643
expanded paths	11,776
placement paths	1,664
full tree paths	87,381
generated poses	108
verified poses	3

Table 3.2: *Simulation 2: Summary of the interpretation tree.*

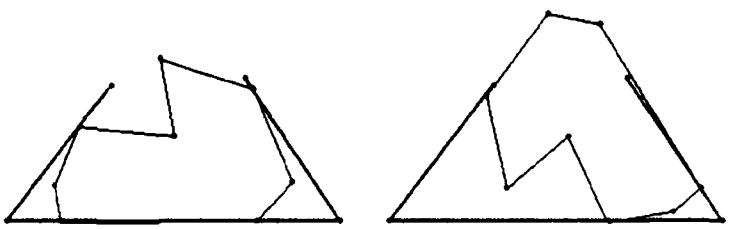


Figure 3.15: *Simulation 2: generated and verified poses. The first pose corresponds to the object's position. One other consistent pose was also found.*

Simulation 3: For this simulation, a two-jointed hand was used again. This object had four vertices. Table 3.3 summarizes the simulation results. As in a previous example, because the object is rather small, a significant percentage of the tree paths were examined. Later examples will better show the power of the pruning operations. Figure 3.17 shows the two poses that were generated and accepted. Figure 3.18 shows the poses that were generated and rejected. All rejected poses relied on the joint torque test, as their geometry is consistent with the hand shape.

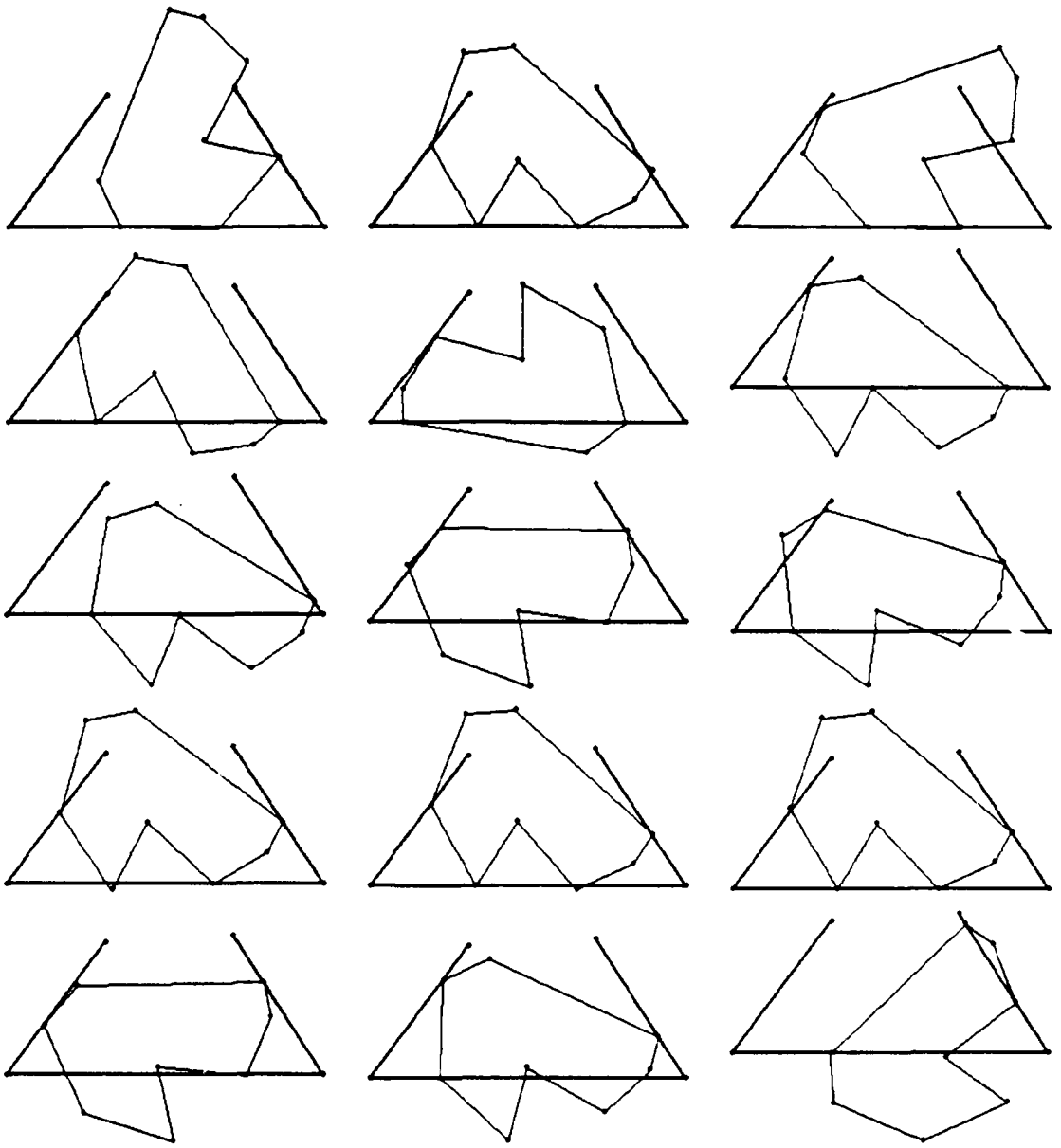


Figure 3.16: *Simulation 2: generated and rejected poses. A selection of poses from the 105 rejected ones are shown here. Note that the upper-left pose was rejected by application of the joint torque constraint.*

vertices	4
finger segments	3
expanded nodes	211
expanded paths	146
placement paths	46
full tree paths	341
generated poses	4
verified poses	2

Table 3.3: *Simulation 3: Summary of the interpretation tree.*

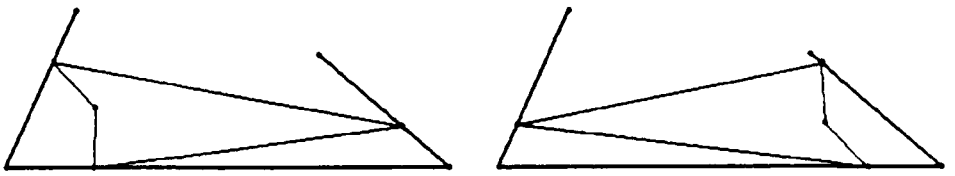


Figure 3.17: *Simulation 3: generated and verified poses. The left pose corresponds to the object's position. The right pose is also totally consistent with the data.*

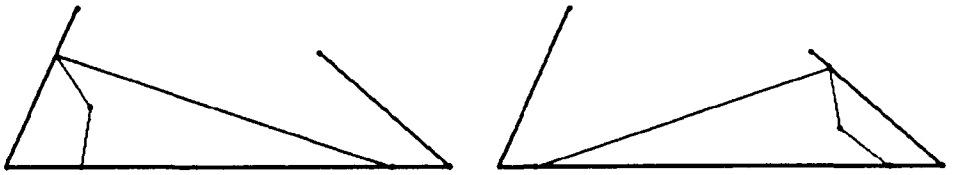


Figure 3.18: *Simulation 3: generated and rejected poses. The entire set of generated and rejected poses are shown. All these poses were rejected by applying the joint torque constraint.*

vertices	8
finger segments	3
expanded nodes	9,121
expanded paths	5,776
placement paths	1,066
full tree paths	87,381
generated poses	58
verified poses	2

Table 3.4: *Simulation 4: Summary of the interpretation tree.*

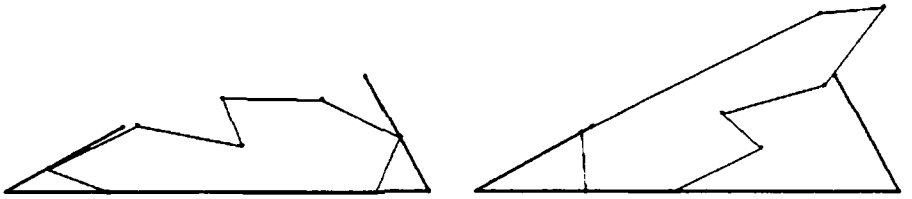


Figure 3.19: *Simulation 4: generated and verified poses. The right pose corresponds to the object's position. The left pose is also totally consistent with the data.*

Simulation 4: For this simulation, a two-jointed hand was again used. This object had eight vertices. Table 3.4 summarizes the simulation results. Two object poses were consistent with the data, and were accepted by both verification tests, as shown in Figure 3.19. Figure 3.20 shows the two poses that were generated but rejected.

Simulation 5: For this simulation, a four-jointed hand was used. The object had five vertices. Table 3.5 summarizes the experiment's results. The object's pose was correctly recovered by the algorithm, as shown in Figure 3.21. Some of the verified but rejected poses are shown in Figure 3.22.

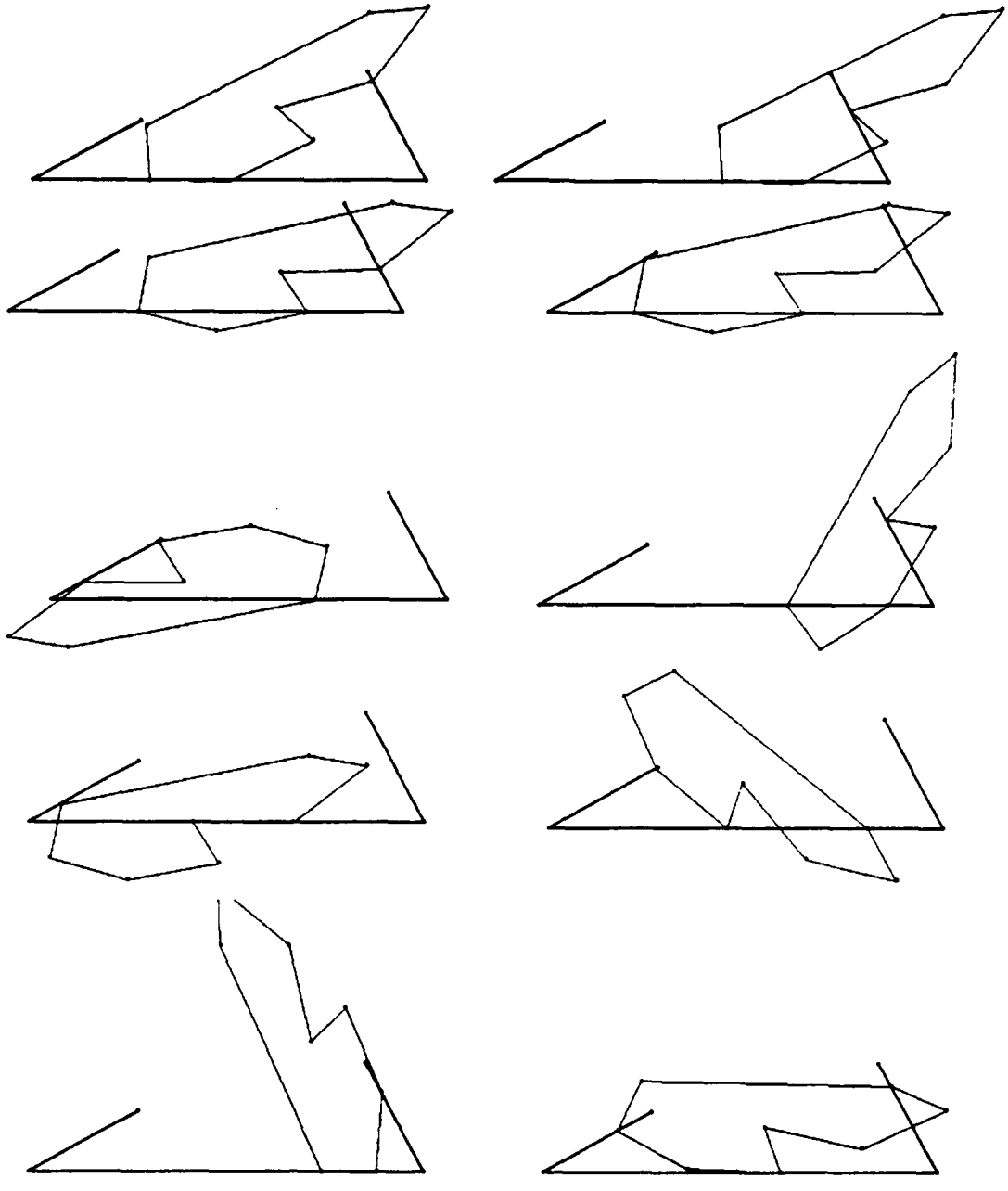


Figure 3.20: *Simulation 4: generated and rejected poses. A sample of the 56 rejected poses are shown.*

vertices	5
finger segments	5
expanded nodes	1,381
expanded paths	972
placement paths	360
full tree paths	9,331
generated poses	5
verified poses	1

Table 3.5: *Simulation 5: Summary of the interpretation tree.*

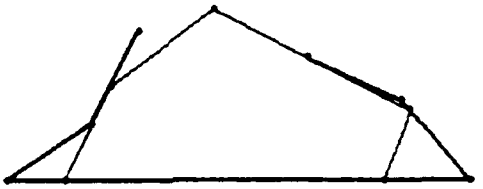


Figure 3.21: *Simulation 5: generated and verified pose. The correct object pose was recovered by the algorithm.*

vertices	6
finger segments	7
expanded nodes	7,791
expanded paths	5,432
placement paths	1,858
full tree paths	299,593
generated poses	32
verified poses	1

Table 3.6: *Simulation 6: Summary of the interpretation tree.*

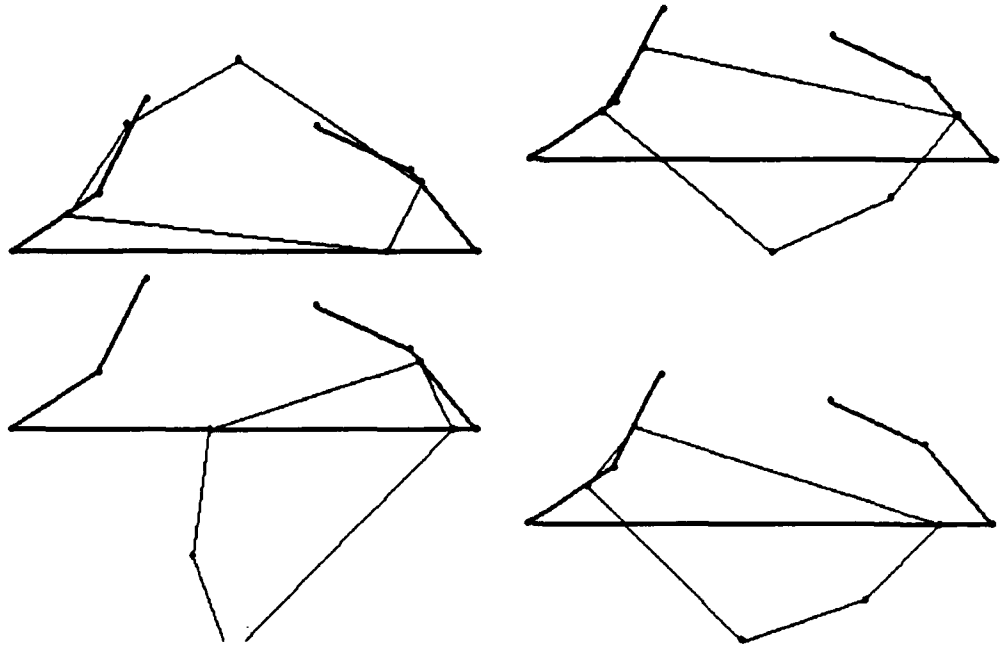


Figure 3.22: *Simulation 5: generated and rejected poses. Some of the rejected poses are shown here.*

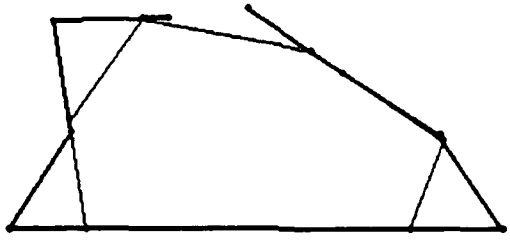


Figure 3.23: *Simulation 6: generated and verified pose.*

Simulation 6: For this simulation, a six-jointed hand was used. The object had six vertices. Table 3.6 summarizes the results. In this case, the larger number of joints and vertices resulted in a potentially large tree size. Nonetheless, only a small number of the full tree's paths were examined. In addition, the number of generated poses was small. This trial provides a good example of how effective this method is for reducing the expensive computations required for finding potential pose candidates. Just one

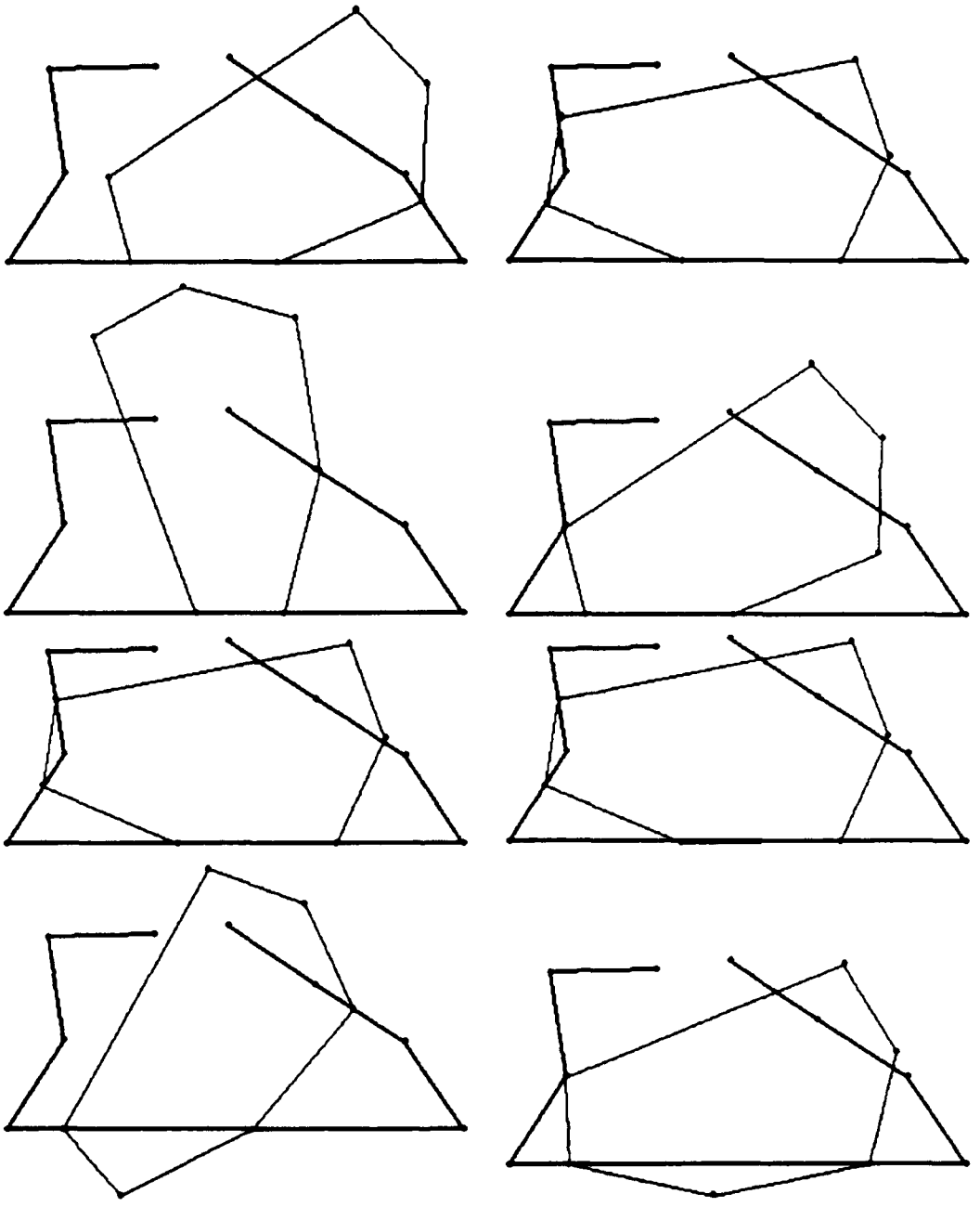


Figure 3.24: *Simulation 6: generated and rejected poses. A sample of some of the 31 rejected poses are shown.*

vertices	9
finger segments	9
expanded nodes	84,844
expanded paths	48,406
placement paths	8,746
full tree paths	1,111,111,111
generated poses	158
verified poses	2

Table 3.7: *Simulation 7: Summary of the interpretation tree.*

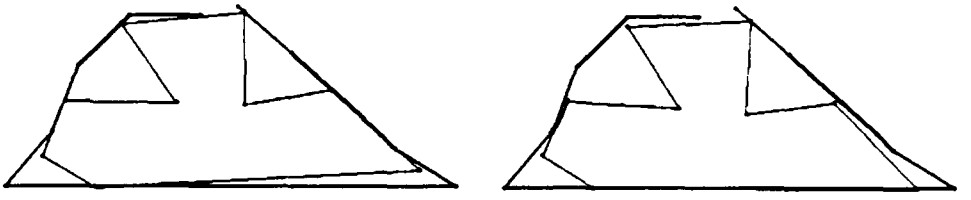


Figure 3.25: *Simulation 7: generated and verified poses. The pose on the left corresponds to the object's actual position.*

pose was generated and verified, as shown in Figure 3.23. Figure 3.24 shows a few of the rejected poses.

Simulation 7: For this simulation, an eight-jointed hand was used. The object had nine vertices. Notice, from Table 3.7, that while the full interpretation tree is huge, the number of consistent paths that were found remained small. Two of the generated poses were verified, as shown in Figure 3.25. The first of those poses corresponds to the object's grasped position. A sample from the 156 generated and rejected poses are shown in Figure 3.26.

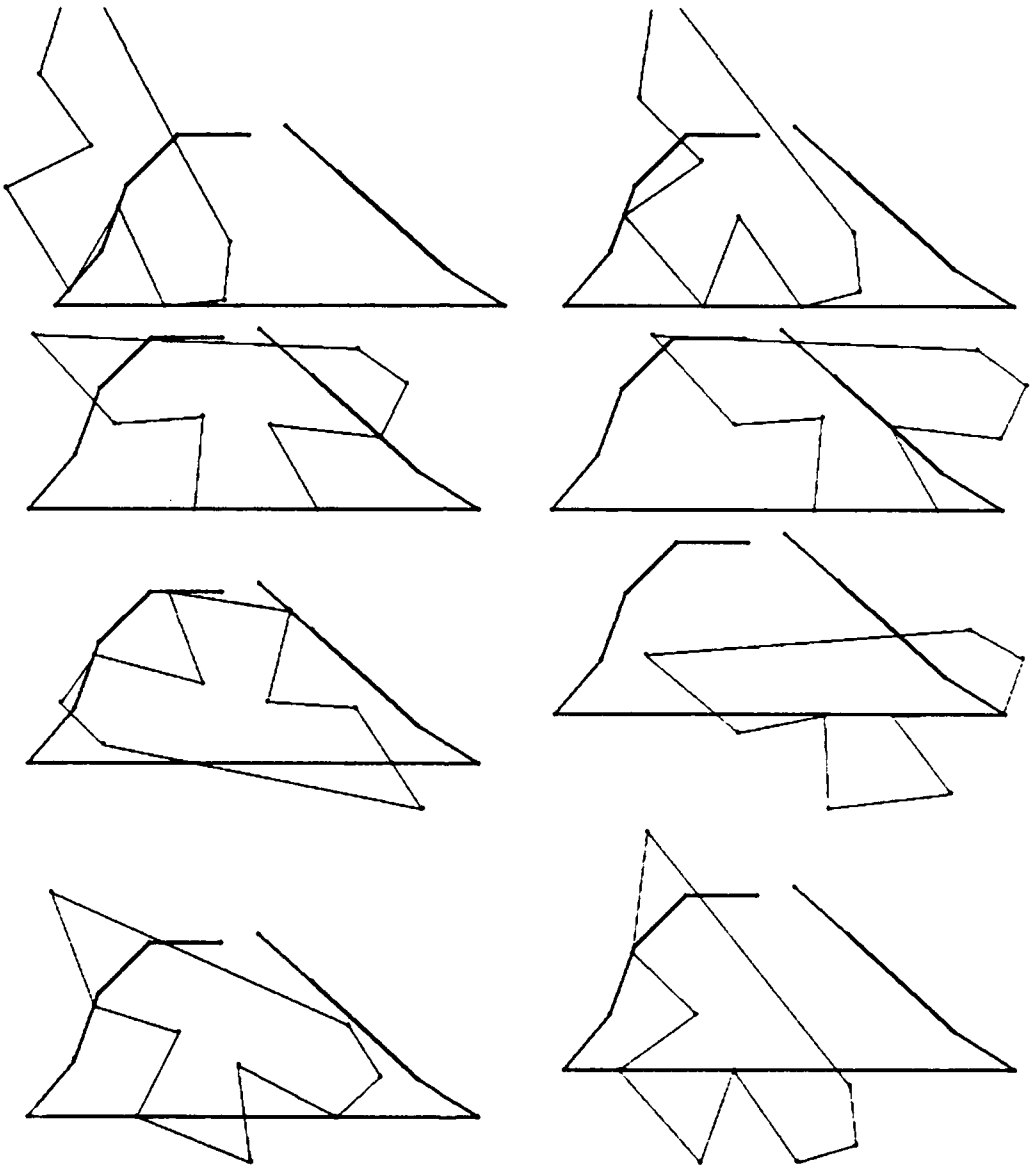


Figure 3.26: *Simulation 7: generated and rejected poses. A sample of some of the rejected candidates.*

vertices	9
finger segments	9
expanded nodes	137,871
expanded paths	84,717
placement paths	10,485
full tree paths	1,111,111,111
generated poses	250
verified poses	5

Table 3.8: *Simulation 8: Summary of the interpretation tree.*

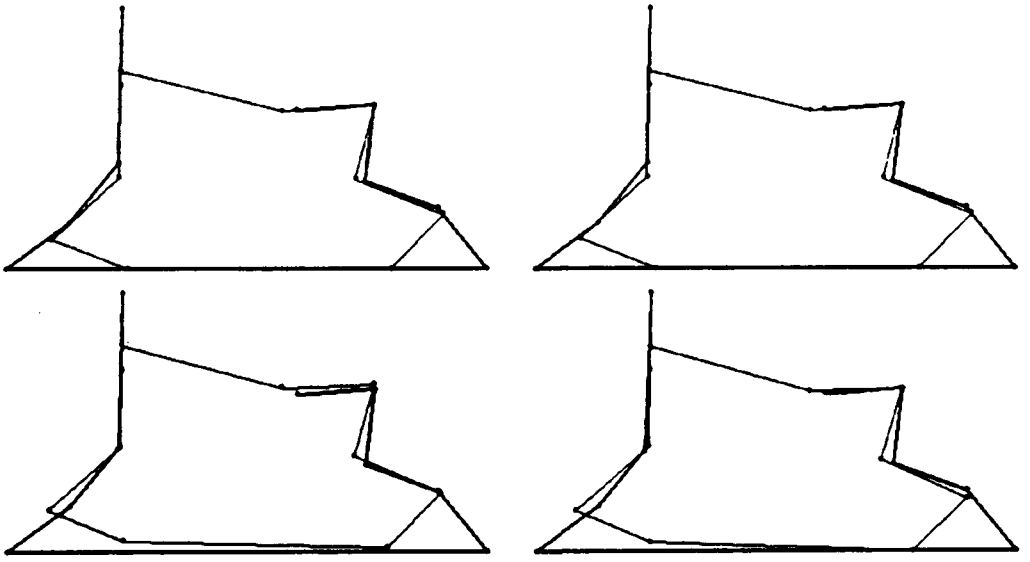


Figure 3.27: *Simulation 8: generated and verified poses. The pose in the upper left corresponds to the object's actual position. The other poses are close verification matches.*

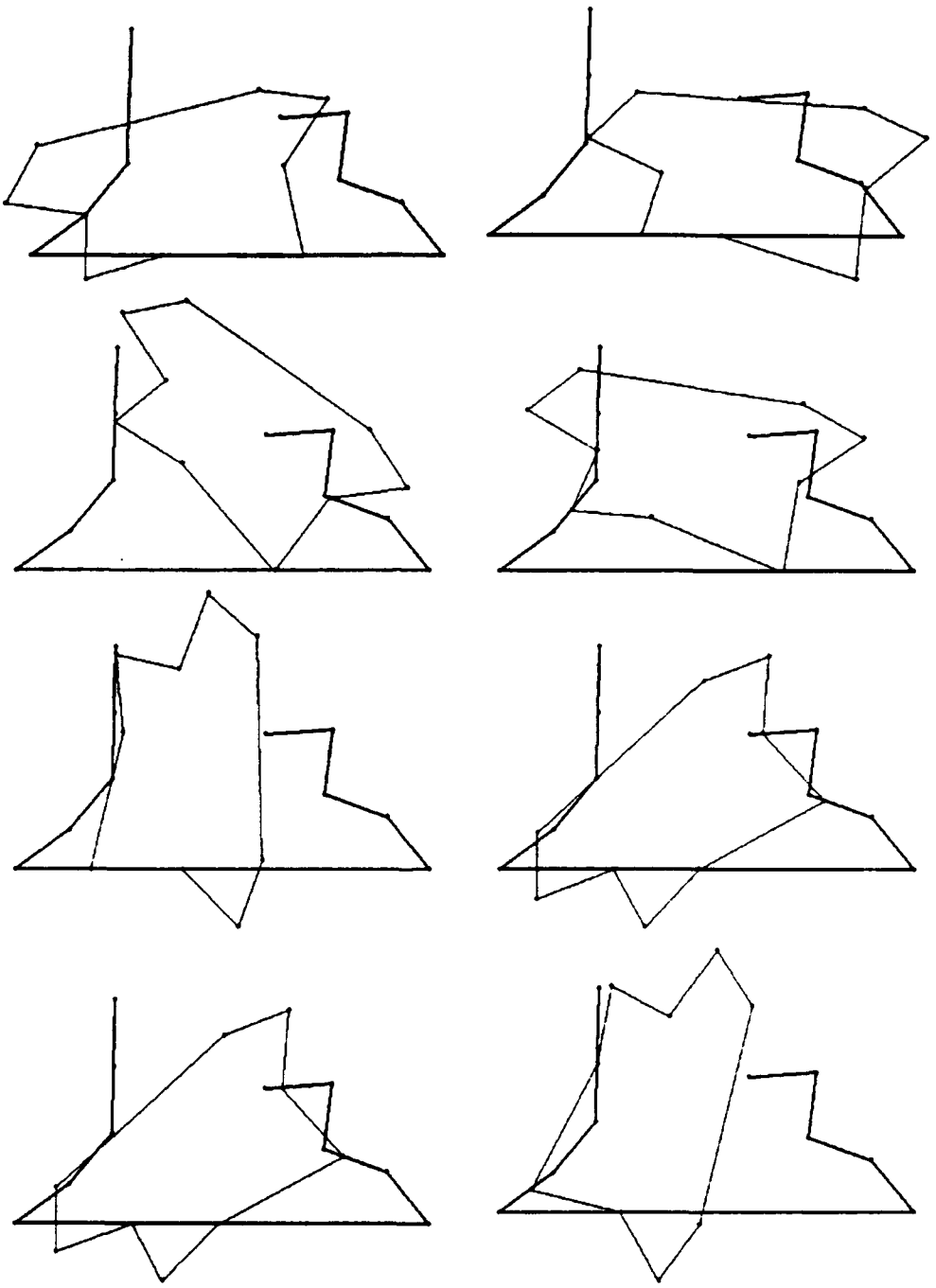


Figure 3.28: *Simulation 8: generated and rejected poses. A sample of some of the 245 rejected poses are shown.*

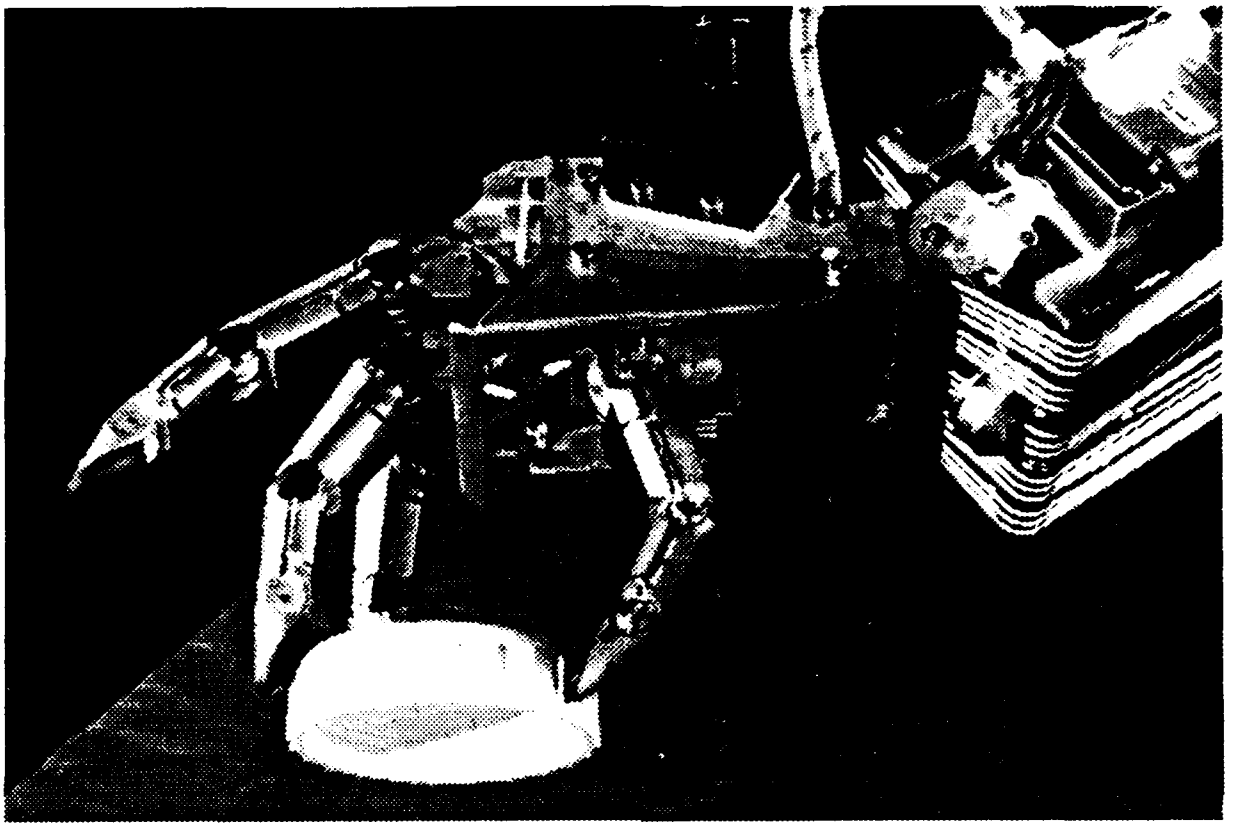


Figure 3.29: *Photograph of the Utah-MIT hand.*

Simulation 8: For this simulation, an eight-jointed hand was used. The object had nine vertices. Notice, from Table 3.8, that the number of consistent paths that were found remains small. In this run, more tolerance was allowed in the verification stage, resulting in a few additional matches. The upper-left pose corresponds to the object's position when it was grasped. Figure 3.28 shows some of the rejected poses. A total of 250 poses candidates were generated in this run.

3.6 Experiments

The pose determination method described in this chapter was tested on the Utah-MIT hand (Jacobsen et al. [57]), using polyhedral objects. A photograph of the hand is shown in Figure 3.29. In general, the experiments confirmed the results obtained from the simulations. The constraint-based pose determination algorithm usually found the object's pose, or at worst found a small set of consistent poses which included the correct pose. This chapter describes the experimental procedures used, and presents a few of the trials.

Since the two dimensional recognition algorithm was used, the test objects were symmetric along the grasping axis. The illustrations that follow are cross sections through this axis. For simplicity, the objects were oriented in a manner to facilitate easy grasping. The three dimension case, using the added constraint that the object were resting on a table-top, was not performed due to limitations in the experimental hardware.

To facilitate the joint torque constraint, the hand was programmed to close on the objects by applying a fixed torque to all its joints. This commanded the hand to wrap around the object with all joints moving forward until contacts were made, or until a joint limit was reached. When the grasp completed, the joint angles were obtained and passed to the recognition module. See Appendix A for a detailed description of the Utah-MIT hand setup, its computational architecture, and its interface to the recognition system.

Trial 1: An object grasped by the Utah-MIT hand is shown in Figure 3.30, along with the position of the hand's fingers that resulted from a grasp of that object. The view of the hand is a cross-section of its xy plane, where the thumb is to the left and the middle finger is on the right (see Narasimhan [74]). Note that joint one of the thumb, the proximal joint, is elevated above the palm plane. Both the thumb and fingers have three joints that move in the xy plane.

For this grasp, a total of 17 possible placements of the object were found prior to the

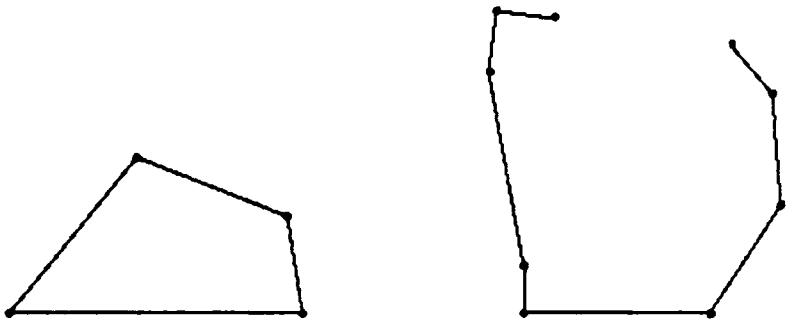


Figure 3.30: *Trial 1: grasped object. The object and the hand shape.*

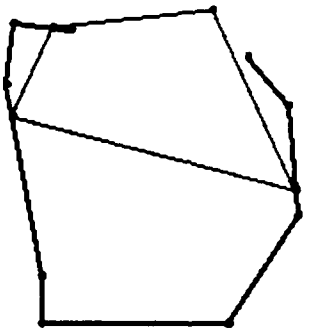


Figure 3.31: *Trial 1: generated and verified pose. Only the actual grasp of the object was generated and verified by the algorithm.*

verification stage. When the verifier applied the intersection and torque constraints, all of the incorrect poses were eliminated, leaving just the correct pose. Figure 3.31 shows the correct pose, as found by the algorithm, while Figure 3.32 shows the object poses that were eliminated by the verifier.

Trial 2: A second object grasped by the Utah-MIT hand is shown in Figure 3.33. Again, the system obtained the correct pose of the object, as shown in Figure 3.34. In this case, however, the verifier accepted two poses that were not correct, as shown in Figure 3.35. As can be seen in the figures, the verified but incorrect poses have symmetries that allowed the object to fit into the hand and still meet all required constraints. Six

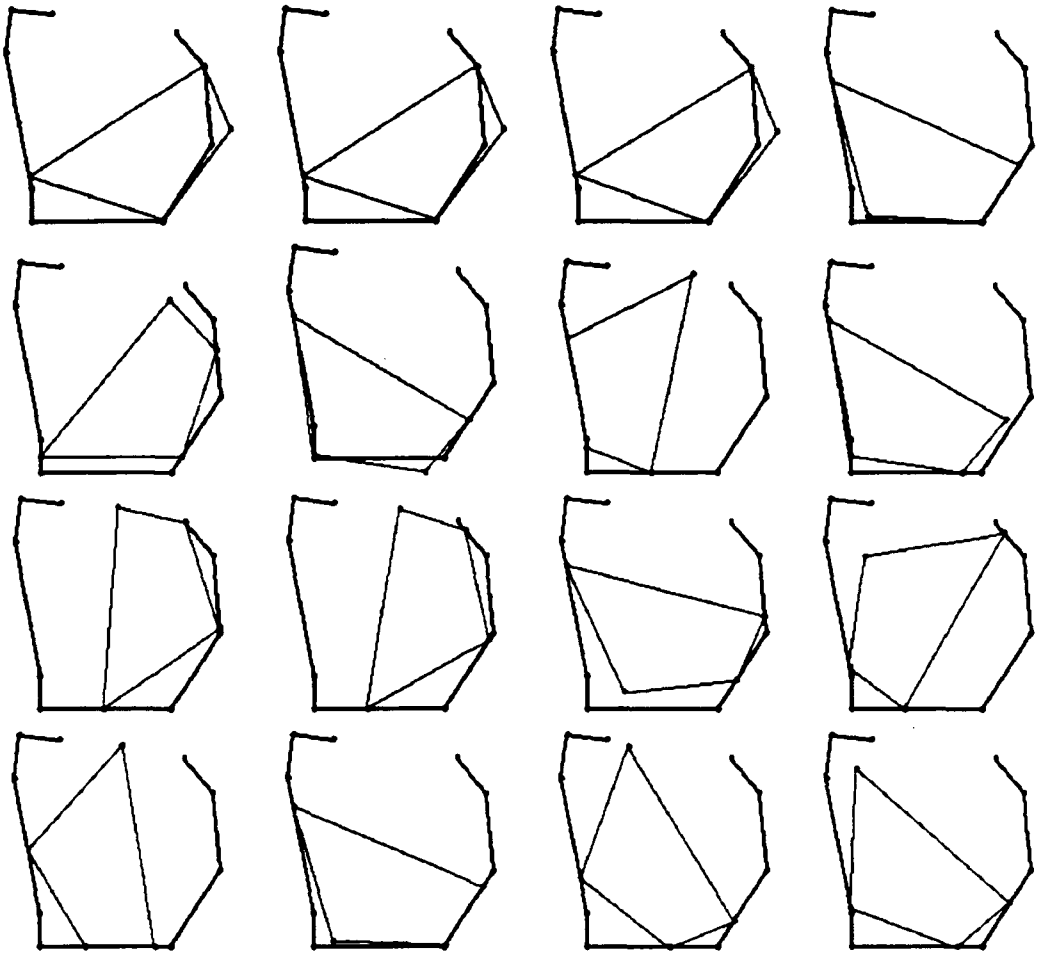


Figure 3.32: *Trial 1: generated and rejected poses. These poses were found by the generator, and rejected by the verifier using either the joint torque of intersection constraints.*

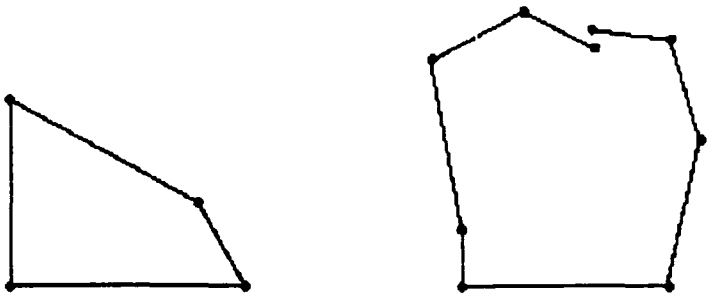


Figure 3.33: *Trial 2: grasped object. The object and the hand shape.*

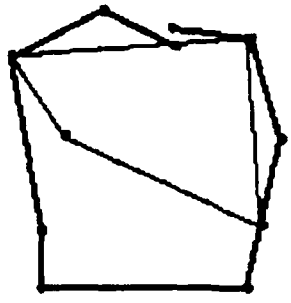


Figure 3.34: *Trial 2: generated and verified pose. The actual pose of the object was found by the algorithm. Note that two other poses that were entirely consistent with the constraints were also found.*

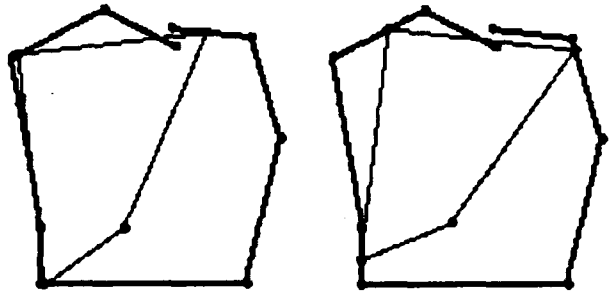


Figure 3.35: *Trial 2: other generated and verified poses. These poses were also found by the algorithm. Though they are consistent with all constraints, they do not correspond to the actual pose of the object.*

additional poses were also found, shown in Figure 3.36. These, however, were eliminated by the verifier.

3.7 Simulations in Three Dimensions

While the method presented in this chapter was described and implemented in two dimensions, extensions to three dimensions are possible. The basic approach would remain the same: assign object features to finger edges, using an interpretation tree to guide the search. For two dimensions, edge-edge and edge-vertex placements are considered when building the tree. When a set of assignments provide three independent

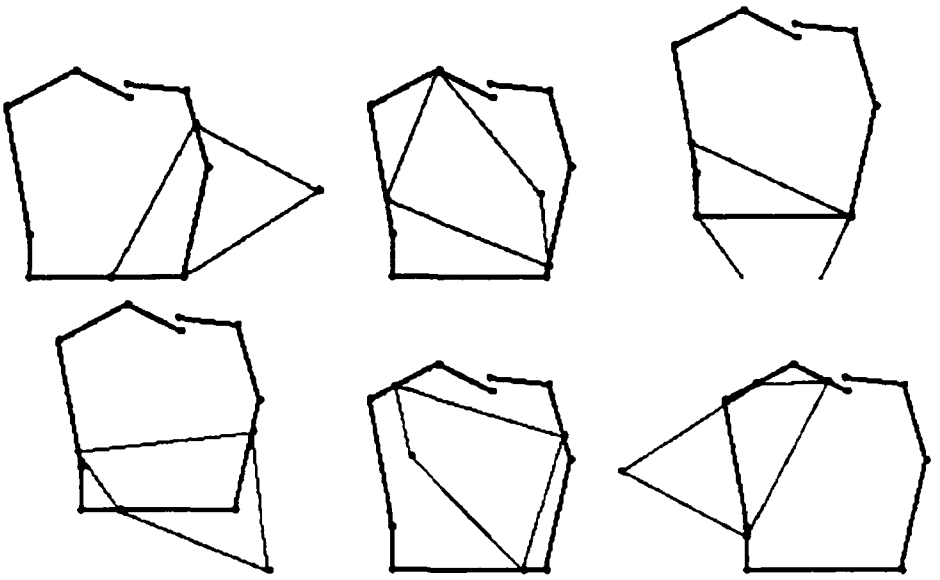


Figure 3.36: *Trial 2: generated and rejected poses. These poses were found by the generator, and rejected by the verifier using either the joint torque of intersection constraints.*

constraints, it is possible to solve for the object's position. For three dimensions, face-face, face-edge, face-vertex, and edge-edge placements can be considered. When a set of assignments provides six independent constraints, the position of the object can be found.

An important question to ask is if the constraints that are used, the hand shape and grasp acquisition strategy, are enough for problems in three dimensions. To gain insight on this, a set of simulations were performed. The hand's fingers are assumed to be modeled as single edge segments. Objects were modeled as a set of edge segments. With this model, edge-edge contacts between the hand and an object are the most common to occur. Grasps of objects were simulated, giving a set of finger edges, some of which are in contact with the object.

A three-dimensional version of the distance constraint was used to expand an interpretation tree. The tree is organized around hypothesized assignments of object edges to finger edges. A new assignment is tested to insure that its parent finger-object edge

assignment is compatible with the new finger-object edge assignment. If it is not, the assignment is pruned. Note that only the most basic distance compatibility test was performed. Range propagation between assignments in the path was not implemented. This makes the constraint significantly less powerful than it would otherwise be in a complete implementation.

Initial results from the simulations indicated that using the basic distance constraint alone was inadequate. The portion of the interpretation tree that was expanded was too large, and the number of consistent candidates that were generated was prohibitive. Perhaps this is not surprising, as the combinatorics for three dimensions is significantly larger than for two dimensions. While an implementation of range propagation techniques was not performed, it is thought that it would help significantly. Nonetheless, additional pruning of the tree is clearly necessary.

In previous sections, the grasp acquisition strategy was used simply as a verification test. Instead, it is possible to use the grasping strategy in the pose generation stage. For example, by using a move-until-contact strategy one can determine the links of the hand that are in contact with the grasped object. Here is how such a procedure would work. Each finger is rolled forward, from distal to proximal, one joint at a time. When contact is detected by monitoring the joint torque sensors, the next joint in the chain is rolled forward. If that joint can move, the contact was made on the previous link. If the joint cannot move, the contact is somewhere up the chain, and will be found latter. This process is repeated for all joints in the finger.

With knowledge of the finger segments that are in contact with the grasped object, a substantial search reduction is possible. In particular, there are two benefits. First, the number of finger segments that are considered in the tree is greatly reduced. Second, the no-contact link from the tree is eliminated. The simulations described below use this additional source of information, and as will be seen, the results are promising.

For each of the simulations, a table summarizing the computations performed is

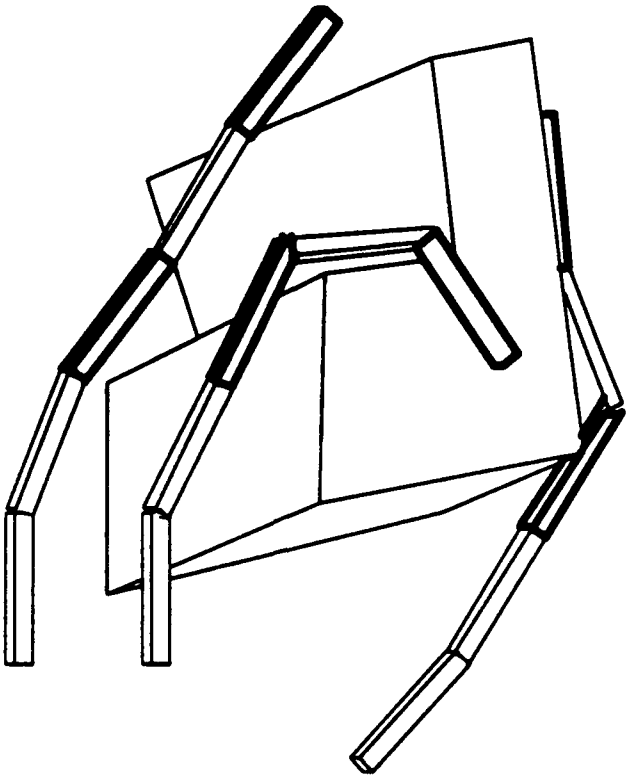


Figure 3.37: *Three dimensions trial 1: object and grasp.*

presented. The terms used in these tables are slightly different from those used in the previous simulations section, and are described here:

- *Object edges* are the number of edges in the object.
- *Finger edges* are the number of finger edges (links), including the palm.
- *Expanded nodes* are the number of nodes in the interpretation tree that were examined and that had at least one expanded child.
- *Pruned nodes* are the number of nodes in the interpretation tree that were examined and that had no children expanded.
- *Placement paths* are the number of paths in the tree that were generated.

object edges	18
finger edges	6
full tree paths	34,012,224

	finger length		
	100%	80%	60%
expanded nodes	28,318	14,208	3,657
pruned nodes	250,071	149,193	47,834
placement paths	8,361	2,575	67

Table 3.9: *Three dimensions trial 1: summary of the interpretation tree.*

Trial 1: The grasped object is shown in Figure 3.37. Note that the highlighted finger edges are the ones that are in contact with the object. The results for this simulation are summarized in Table 3.9. Here, the columns indicate the percent of the full length of each finger segment that was used when generating the tree. The column labeled 100% gives the results for the full finger segments, as shown in Figure 3.37. For lower percentages, the finger segments are reduced in length by the given amount. Essentially, the endpoints of the segment are adjusted along the line to reduce the segment length, while preserving the contact point between the finger and object edge. By reducing the finger segment length, the distance constraint becomes more powerful. This helps, for example, investigate how much additional recognition power would be present in a hand of similar total finger length, but with more finger links.

For this trial, the simulation results are quite promising. The portion of the tree expanded is manageable, and the total number of poses that need to be generated is small enough to be feasible. Note that by reducing the finger segment length, a significant reduction in tree size was achieved. The results shown here are among the best that

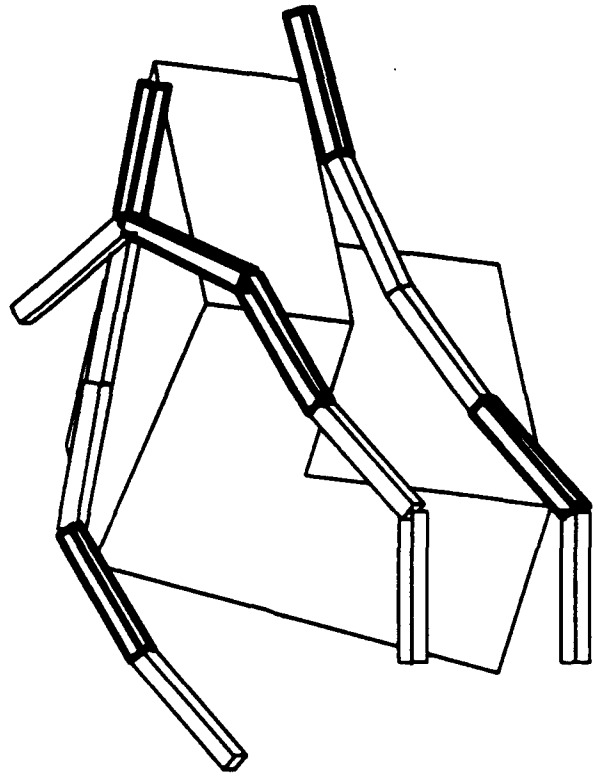


Figure 3.38: *Three dimensions trial 2: object and grasp.*

were achieved during a number of simulations.

Trial 2: Not all simulations produced results as good as those shown in the previous example. An object similar to the one used in that example was grasped, as shown in Figure 3.38. The object had slightly different dimensions and a different orientation from the previous object. The simulation results are summarized in Table 3.10.

For this trial, a rather large number of nodes in the tree were explored for the case of 100% edge length. In addition, a large number of consistent paths of length 6 were found. As the edge length was reduced in size, as previously described, the results became more promising. For a 60% reduction, a manageable 6,740 candidates were generated.

object edges	18
finger edges	6
full tree paths	34,012,224

	finger length		
	100%	80%	60%
expanded nodes	196,960	133,103	24,536
pruned nodes	1,014,083	733,765	212,130
placement paths	104,798	67,279	6,740

Table 3.10: *Three dimensions trial 2: summary of the interpretation tree.*

3.8 Summary and Discussion

This chapter described a constraint-based method for localizing objects grasped by a hand. The class of recognition problems discussed are important for utilizing robotic hands in manipulation tasks. In general, the experiments and simulations confirm that it is usually possible to unambiguously identify the pose of an object grasped by a hand using just joint angle and torque data. An unambiguous recognition is most likely when the object has many vertices that are different distances apart from each other. For objects with a great degree of symmetry, like a square, it is impossible to distinguish certain orientations. In the case of the square, of course, there is nothing that distinguishes 90 degree rotations, so any localization scheme would suffer from the same failing.

The results obtained are very promising. Data gathered from actual objects grasped by the Utah-MIT hand indicate that pose determination and small set object recognition are feasible using joint sensor values obtained from just two fingers. The experiments utilized only 6 of 16 joint angle readings available from the hand. The performance of

the recognizer on this data was limited by the lack of a good model of the hand, and apparently not by the limited data. The surfaces of the hand are all different sizes and shapes, necessitating a complex model. At the current time we do not have such a model available.

Simulations of the tree pruning portion of the algorithm in three dimensions were presented. The results again indicate that hand shape and the grasp acquisition strategy have considerable recognition power. The combinatorics in three dimensions, however, are of concern. The added pruning power that reducing the finger link length has on the problem indicates that addition contact location information, perhaps from tactile sensors or additional finger links, may be necessary to make the method's performance acceptable.

Memory-Based Pose Recognition

Chapter 4

4.1 Introduction

This chapter presents a pose determination strategy based on a table-lookup operation from a memory. The memory is filled with hand shapes that result from grasping particular objects. Estimates of an object pose are obtained by matching a hand's shape to the experiences that have been stored in the memory. Because the lookup operation is fast, determination of poses that have been encountered before is fast.

The determination method uses what is called the *grasp acquisition strategy* constraint. This constraint is simply the information inherent in the knowledge of how the hand was programmed to acquire objects. The pose determination memory is filled using this constraint. An interesting result from this chapter is that by using the grasp acquisition strategy as a constraint, the memory size is reduced and the number of ambiguous poses in each memory entry is reduced. Intuitively, a particular grasp acquisition strategy limits the hand shapes that can occur, which limits the number of ways an object can be grasped. This helps makes the use of a memory feasible.

Some compromises are made. The use of memory essentially trades off time for space. Because configuration space is tessellated, pose accuracy is proportional to the tessellation granularity. Techniques for improving the accuracy by using interpolation are discussed. Additional sensor information can also be used to improve pose estimates. Chapter 5 discusses such a technique, based on data obtained from fingertip force sensors.

The term *hand shape* is used throughout this chapter. A hand's shape is obtained from its kinematic model, and from its joint positions. In this chapter, the term *joint angles* is used interchangeably with the term *hand shape*.

4.1.1 Relevance of this Problem

The previous chapter developed a constraint-based method for obtaining object pose estimates. While the method avoids the potential combinatorial explosion of searching the entire pose interpretation space, it is not fast enough to qualify as being real-time. This chapter examines the use of a memory to speed the determination process. Faster pose determination is desirable in all cases. When a real-time pose determination is required, the faster the method, the better.

4.1.2 Why Use Grasp Acquisition Strategy Constraints?

The power behind this method lies in the observation that while, from a mechanical standpoint, a hand has a large configuration space, its high level control and planning strategies usually limit the space's size. Consider the configuration space for the Utah-MIT hand. The hand has four fingers, each with four degrees of freedom. Thus, the space of possible hand positions has 16 dimensions. The less complex Salisbury hand, with three fingers, each with three degrees of freedom, has a smaller space of possible grasps, though the space is still huge. A grasp acquisition strategy provides a way to limit the number of shapes of the hand that can occur.

Whenever a strategy is used to limit the number of potential hand shapes, it is

possible that useful shapes will be omitted. A useful hand shape is one that is required for grasping an object in a particular configuration. Thus, while a grasp acquisition strategy is useful for reducing the combinatorics of the hand configurations, it may provide sub-optimal grasping performance. One way to overcome this is by using a set of grasping strategies, each designed for a particular situation.

The observation that a grasp acquisition strategy greatly limits possible grasps can be exploited for determination. For each particular position of an object in a hand's workspace, a simulator for the grasp acquisition strategy can be run to determine the grasp that will result if the strategy worked. The hand shapes found in this process are the entire set of shapes that occur when grasping the object. As will be seen, the determination method described in this chapter is based on this principle.

4.1.3 Chapter Overview

The following sections will show how memory-lookup operations can be used for pose determination. Section 4.2 provides an overview of previous work. Section 4.3 outlines the assumptions required for the solution given. The approach is discussed in Section 4.4. Section 4.5 discusses the experimental setup and results. Section 4.6 presents conclusions.

4.2 Related Work

The work in this chapter is related to memory-based schemes for learning and modeling. Two approaches can be considered, those that store experiences directly, and those that represent experiences by a set of parameters. Atkeson and Reinkensmeyer [4] provide a good recent review of this field.

When experiences are stored directly, nearest neighbor approaches are used to find similar experiences to a new event. Interpolation from limited examples is also possible. Methods for this are reviewed by Barnhill [6] and Sabin [89]. Local models can be

formed between the nearest neighbors for each access into the memory. See Watson [107], Cover [24], and Shepard [93] for examples of this approach. McLain [71], Stone [101], Franke and Nielson [37] and others use a distance weighted regression to fit polynomial surfaces to data.

Though memory searches can be performed on serial computers, parallel machines make the searches much faster. Stanfill and Waltz [99] learn pronunciation by searching for related experiences using the massively parallel Connection Machine (Hillis [46]).

Connectionist networks, or neural networks, provide an alternative way for representing past experiences. See Rumelhart and McClelland [87] and Hinton [48] for overviews of the field. Essentially, a set of nodes and links are constructed to approximate a function that maps inputs to outputs. The use of the network permits both generalization over the training experiences, and more compact storage of past experiences in a memory.

4.3 Assumptions

The following assumptions are made in this chapter:

- The hand has been modeled.
- The objects have been modeled.
- Either a grasp simulator, or an object pose determiner is available.
- The grasped object is assumed to be in static equilibrium.
- Hand joint angle sensor data is available.

4.4 Approach

This section describes the memory based pose determination approach in detail. The only sensor inputs to the recognizer is the set of hand joint angles. The output from the

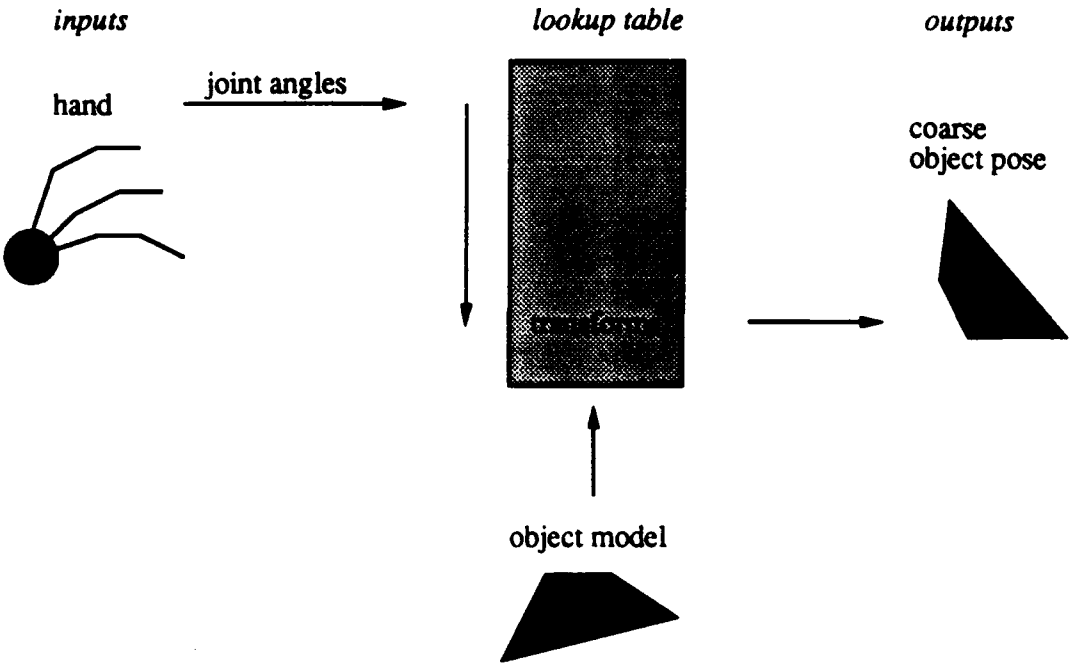


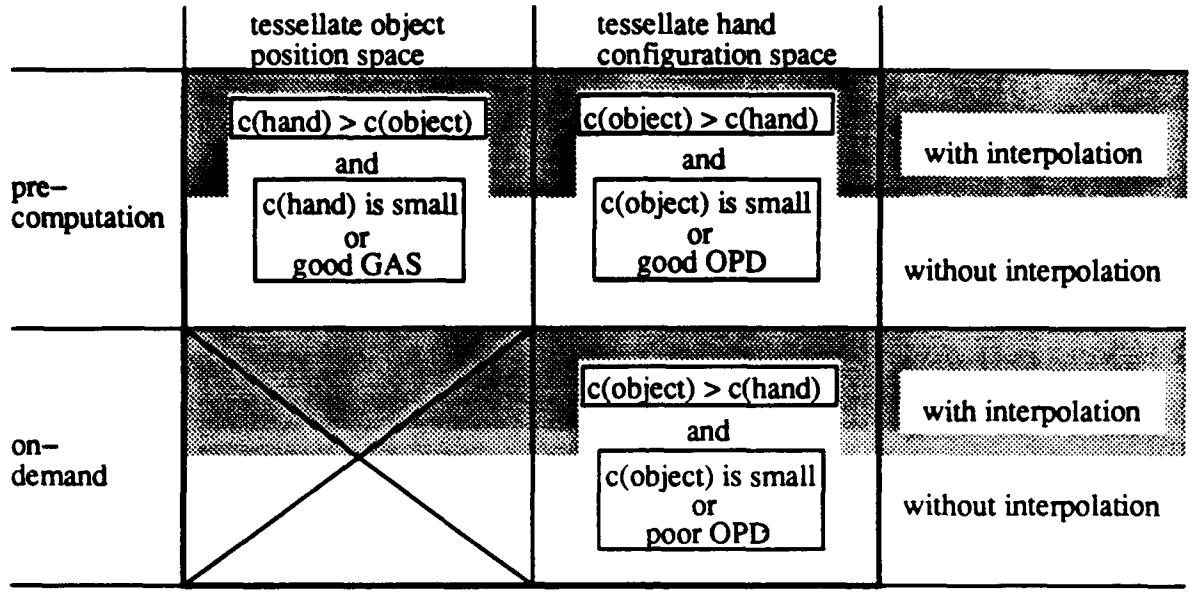
Figure 4.1: *Overview of memory-based pose determination.*

recognizer is a coarse estimate of the object's position in the hand. Figure 4.1 diagrams the method, showing the flow of information from inputs to outputs.

4.4.1 Overview

There are a variety of approaches for organizing, filling, and using the determination memory. The matrix in Figure 4.2 outlines the possibilities that are considered. The memory can be organized around a tessellated object configuration space or around a tessellated hand configuration space. The memory can be filled by pre-computation, prior to determination, or on-demand, during determination. The memory can be used by extracting the best pose match, or by interpolating between a set of close pose matches. Note that one of the approaches, on-demand tessellated object space, is not possible. This will be explained later in this chapter.

More formally, a determination memory is used to approximate a function R that



c(hand)	configurations of the hand
c(object)	configurations of the object
GAS	grasp acquisition simulator
OPD	object pose determiner

Figure 4.2: *Possible memory organizations. A variety of approaches for organizing, filling, and using determination memories are possible. This table illustrates eight potential methods.*

maps hand configurations (shapes) to pose entries:

$$R(C_H) = \{P_1, P_2, \ldots P_n\}, \tag{4.1}$$

where C_H is the configuration of hand H, and P_i is a pose. It is possible for $R(C_H) = \emptyset$. A pose entry P_i contains:

1. A pointer to the object model.
2. A transform from the object model O to an instance of the object $O(\bar{x})$ at position $\bar{x}$.
3. The hand configuration, C_H.

The next sections will examine how this function is approximated using a memory.

4.4.2 Organizing the Memory

The determination memory can be organized around a tessellated object configuration space or a tessellated hand configuration space. For tessellated object configuration organizations, memory locations are computed by finding the hand shape that result from grasping the object at a particular location. For tessellated hand configuration organizations, memory locations are computed by finding the object poses that result from a particular hand shape.

The choice of organization to use is in part a function of the relative size of the tessellated spaces. For two-dimensional problems, a tessellated object space has three dimensions. For three-dimensional problems, the space has six dimensions. For tessellated hand configurations, the dimensionality of the space is equal to the number of degrees of freedom of the hand.

Potentially, the space of possible hand configurations will be much larger than the space of object positions. However, the grasp acquisition strategy constraint can dramatically reduce the effective size of the hand configuration space. Potentially, it can be reduced to one degree of freedom, even for problems in three dimensions. A hand being used like a parallel jaw gripper is an example of this.

Tessellated Object Space

A grasp simulator is used to compute memory locations for a tessellated object space. The grasp simulator G, when given an instance of an object O, finds the hand configuration C_H that would result from the grasp:

$$G(H, S, O(\bar{x})) = C_H, \tag{4.2}$$

where $O(\bar{x})$ is an instance of object O at location $\bar{x}$, and S is the grasping strategy being used.

Tessellated Hand Space

An object pose determiner is used to compute memory locations for a tessellated hand space. When given a hand configuration C_H, a pose determiner finds the set of object poses that are consistent with the hand shape. This can be thought of as the inverse of Equation 4.2:

$$G^{-1}(C_H) = \{O(\bar{x}_1), O(\bar{x}_2), \ldots O(\bar{x}_n)\}. \tag{4.3}$$

The constraint-based pose determination approach described in Chapter 3 provides an algorithm for computing Equation 4.3. The method finds all object poses that are consistent with a particular hand shape, assuming certain types of contacts and termination predicates. Unlike with grasp simulators, object pose determiners make fewer assumptions about the world. For example no assumptions need be made about whether the object was stationary during the grasping operation. This advantage makes it easier to fill the memory accurately.

Simulation Issues

What are the requirements for a good grasp simulator? Most importantly, the simulated gripping process should correspond to how the hand would behave when actually grasping the object. If the simulator's output differs from reality significantly, the resulting memory entry would be meaningless. One of the most difficult factors to consider is motion of the object during the grasping process. It is hard for a simulator to predict such motion correctly. As will be seen, an advantage of tessellating the hand configuration space is that it does not require a grasp acquisition simulator.

It is interesting to note the relationship between this type of grasp simulator and a grasp planner. Grasp planners, like grasp simulators, must also consider the factors listed above. The actual grasp and finger motions that a planner generates are assumed to be executable by the hand. Thus, the planner must have some of the same knowledge that a simulator contains, in order to plan realizable motions.

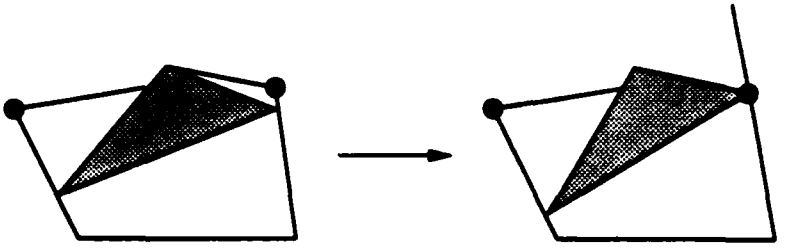


Figure 4.3: *Marginal grasp. A small change in the object's position will cause a large change in the hand's shape. Likewise, grasps are possible where a small change in the hand's shape will cause a large change in the object's pose.*

The grasp acquisition simulator, when applied to an object in a particular pose, must find the resulting hand shape. If the object is out of reach, no grasp will be found. For an object in reach, the resulting grasp can be categorized as follows:

1. A small change in the object's position will cause a small change in the hand's shape, and vice versa.

2. A small change in the object's position will cause a large change in the hand's shape.

3. A small change in the hand's shape will cause a large change in the object's position.

In the first case, the pose is considered to be a good one, and the hand shape is a good indication of the object's pose. For the second two cases, the mapping between hand shape and object pose is not as well defined (see Figure 4.3). Two problems occur with these *marginal* grasps. Coarse tessellations of a space could miss particular configurations, and sensitivity may be poor in the regions around the marginal configurations, depending on the space that was tessellated.

To examine the issue of marginal grasps in more detail, refer to Figure 4.4. Each axis represents the level of tessellation of the labeled space. The plots in the figure can be used

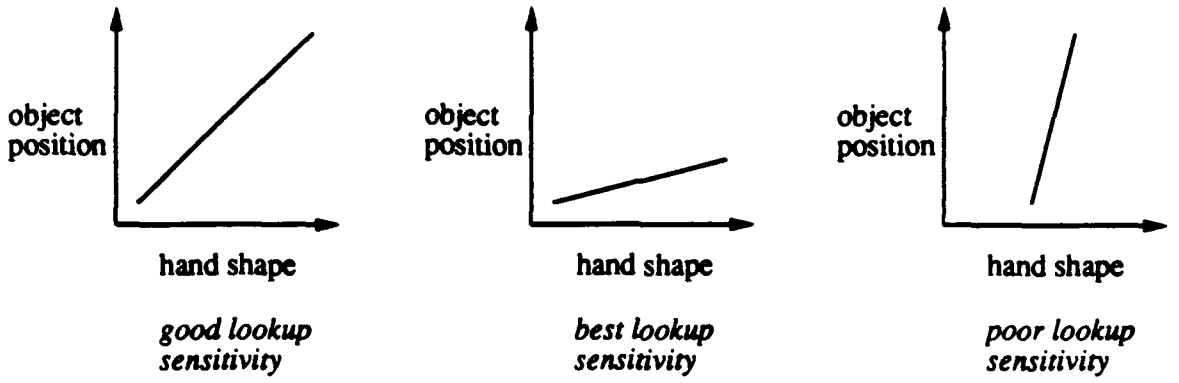


Figure 4.4: *Memory sensitivity. When mapping hand shapes to object poses, three classes of conditioning can be considered. Poor conditioning occurs in the third case, when a small range of hand shapes maps to a large range of object positions.*

to understand when the problem is poorly conditioned. In the first case, the conditioning is good, independent of which space is tessellated. In the second case, the conditioning is excellent when hand space is tessellated. If the object space is tessellated, a sparse sample of hand shapes would result. In the third case, conditioning is unavoidably poor, independent of tessellation considerations. Since a wide range of object positions map to a small range of hand shapes, using joint position sensing for determination cannot work well. Thus, the only case that is of concern is the second case. By tessellating the hand space this conditioning problem is avoided.

4.4.3 Filling the Memory

Both pre-computation and on-demand approaches are considered for deciding when to perform the computations required for filling the memory. If the tessellated space is small enough, it is feasible to pre-compute the memory. This is desirable, as on-line determination becomes a fast, constant-time lookup. If the space is too large, this pre-computation becomes impractical. As an alternative, portions of the memory that are encountered can be computed on-demand. This scheme reduces performance when a

new hand shape is encountered, though the results are then committed to the memory, which speeds future determinations.

A benefit of the on-demand computation is that only the portions of the space that actually occur are computed. This reduces the size of the memory. The use of an on-demand approach can also reduce dependence on the grasp acquisition strategy. If the memory is organized around hand configurations, pre-computation requires knowledge of the hand shapes that are possible, as provided by the grasp acquisition strategy. By performing on-demand computations, this knowledge is no longer needed.

In certain situations, the configurations that an object can assume are limited. The pre-computation approach is especially desirable in this case. For example, in a factory environment a robot may be acquiring a part from an assembly line. The part might have a nominal position, though vibrations and other sources of motion may introduce a certain amount of placement error.

Note that on-demand tessellated object position space is not a possible approach for pose determination. If a new hand shape is encountered, the on-demand approach would require simulating all possible object positions to find the ones that are compatible with the hand's shape. This, in essence, would be a pre-computation of the entire space.

4.4.4 Using the Memory

From an implementation standpoint, various approaches can be used for approximating the mapping function R, including hash tables, content addressable memories, and neural networks. An approach based on hash tables is used for the experiments conducted in this chapter.

Hand configurations are entered into the determination memory using an index, or key. The index is generated from the hand's finger joint angle array. A particular bucket size is used to cluster adjacent joint angles together. The bucket size is selected based on several considerations, including the actual accuracy expected from the joint angle

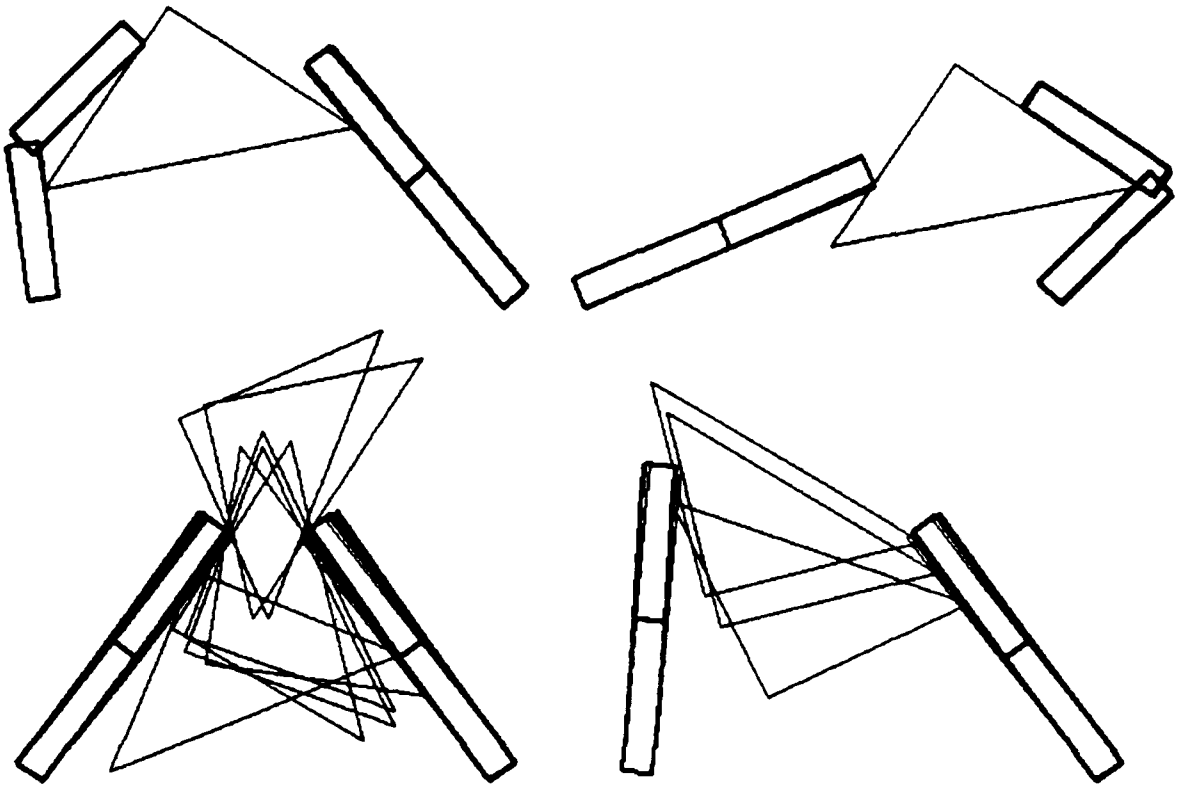


Figure 4.5: *Determination memory entries. The contents of four different determination memory locations are shown here. The upper locations each contain one object pose, and thus uniquely identify the object's location. The lower locations have multiple entries, and thus cannot uniquely identify where the object is position.*

sensors on the hand. Multiple scale indices can be generated, using different bucket sizes.

Figure 4.5 shows the contents of four entries in a typical determination memory. The hand drawn in dark lines is the shape of the entry key. The hands drawn in lighter lines correspond to the objects in the particular entry. The memory entries represented by the upper figures contain just one object pose, and thus uniquely identify the object's pose. The memory entries represented by the lower figures contain multiple entries, and thus do not uniquely identify where the object is positioned.

Thus, a determination memory provides a direct mapping from a set of hand joint

angles to objects and their positions. Determination is achieved simply by generating a key from a hand shape and hashing into the memory. The entry will contain the one or more poses of the object that are consistent with the hand shape.

4.4.5 Memory Interpolation

Determination memories, as previously described, suffer from two related problems. The first problem concerns the lack of a limit on memory size. Finer tessellations result in more memory usage. While it has been argued that useful tessellations have a manageable memory size for certain classes of problems, more efficient use of memory is always desirable. The second problem concerns close, but imperfect, matches. Essentially, a hand shape is indexed into the memory, and close matches are extracted. The pose that corresponds to the best match is selected as the candidate. It is possible to better estimate the pose by interpolating between related memory entries. This section explores both these ideas.

Object grasps are considered to be in related configurations if the object vertex to finger segment assignments are the same. In these related configurations, continuous changes in the object position result in continuous changes in the hand shape. In fact, a particular function exists that maps hand shapes to object poses, for each vertex-segment configuration. Obtaining this function, and a closed form solution for it, is hard. It is also dependent on the particular hand's kinematics. Because of this, a general second order function is assumed, and regression analysis is used to identify the function's coefficients. As an example, consider a two jointed, two fingered hand. There exists a set of functions of the form:

$$\begin{aligned} f_{xc}(\theta_1,\theta_2,\theta_3,\theta_4) &= a_1 + a_2\theta_1 + a_3\theta_2 + a_4\theta_3 + a_5\theta_4 + a_6\theta_1^2 + a_7\theta_2^2 + a_8\theta_3^2 + a_9\theta_4^2 + \\ &\quad a_{10}\theta_1\theta_2 + a_{11}\theta_1\theta_3 + a_{12}\theta_1\theta_4 + a_{13}\theta_2\theta_3 + a_{14}\theta_2\theta_4 + a_{15}\theta_3\theta_4 \\ f_{yc}(\theta_1,\theta_2,\theta_3,\theta_4) &= b_1 + b_2\theta_1 + b_3\theta_2 + b_4\theta_3 + b_5\theta_4 + b_6\theta_1^2 + b_7\theta_2^2 + b_8\theta_3^2 + b_9\theta_4^2 + \\ &\quad b_{10}\theta_1\theta_2 + b_{11}\theta_1\theta_3 + b_{12}\theta_1\theta_4 + b_{13}\theta_2\theta_3 + b_{14}\theta_2\theta_4 + b_{15}\theta_3\theta_4 \end{aligned} \quad (4.4)$$

$$f_{\theta c}(\theta_1, \theta_2, \theta_3, \theta_4) = c_1 + c_2\theta_1 + c_3\theta_2 + c_4\theta_3 + c_5\theta_4 + c_6\theta_1^2 + c_7\theta_2^2 + c_8\theta_3^2 + c_9\theta_4^2 + c_{10}\theta_1\theta_2 + c_{11}\theta_1\theta_3 + c_{12}\theta_1\theta_4 + c_{13}\theta_2\theta_3 + c_{14}\theta_2\theta_4 + c_{15}\theta_3\theta_4$$

where c is the configuration number, θ_i are the hand's joint angles, and $\{a_i, b_i, c_i\}$ are the coefficients to be identified. In general, there will be $3(n^2 - 1)$ coefficients, where n is the number of joints in the hand. Note that there is one function for each of the unknown parameters required to position the object.

Equation 4.4 can be used to both solve the problem of large memory size and to allow interpolation between existing memory entries. All entries in a configuration can be summarized by the a coefficients from the equation, eliminating the need for storing each joint angle to object pose mapping entry separately. This greatly reduces the size of the memory, making it proportional to the number of contact configurations, rather than the size of the tessellated space. The equation, when given a set of joint angles, will return the best estimate of the object's pose that is consistent with the data used to find the a coefficients. This provides the desired interpolation capability.

Each instance of the set of functions shown in Equation 4.4 is valid for a particular set of object vertex to finger segment assignments. For determination, only the hand's joint angles are available. There is no knowledge of the particular contact configuration that has occurred. The question then remains as to which particular equation, from the set of c equations, should be applied. Put another way, a mapping from joint angles to contact configurations is required.

One approach that can be used is to try all c sets of functions. Each set of equations will return a particular object position. For all but one of the sets of equations, the contact configurations will be incorrect, and hence the computed object position is incorrect. To determine which function was the appropriate one to use, the computed object position is simply verified against that hand's shape. If the position gives the contacts between the hand and object that were required for the set of functions, then the pose is accepted. Otherwise, it is rejected.

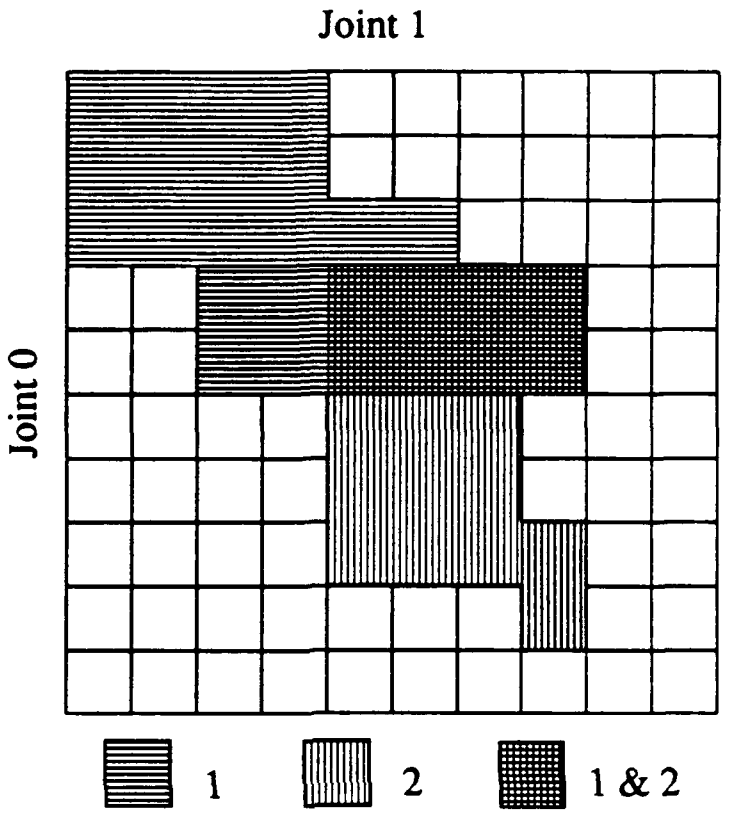


Figure 4.6: *Contact configuration memory. This figure diagrams the coarse memory that maps joint angles to contact configurations, for a hand with two joints. In this example, two contact configurations occur. Each entry contains the one or more contact configurations that occur for the entry's range of joint angle values.*

A more efficient approach that can be used is to build a coarse memory that maps hand shapes to contact configurations, and hence directly to the determination functions (see Figure 4.6). Thus, a memory is built that is indexed by joint angles, and returns sets of determination equation coefficients. This memory plays a role similar to the initial scheme described, where the memory maps joint angles directly to an object position. However, a memory that maps joint angles to contact configuration information can be made more coarse, since it is not used to directly obtain the object's position. Rather, it is used to map to interpolation functions, which then compute the position.

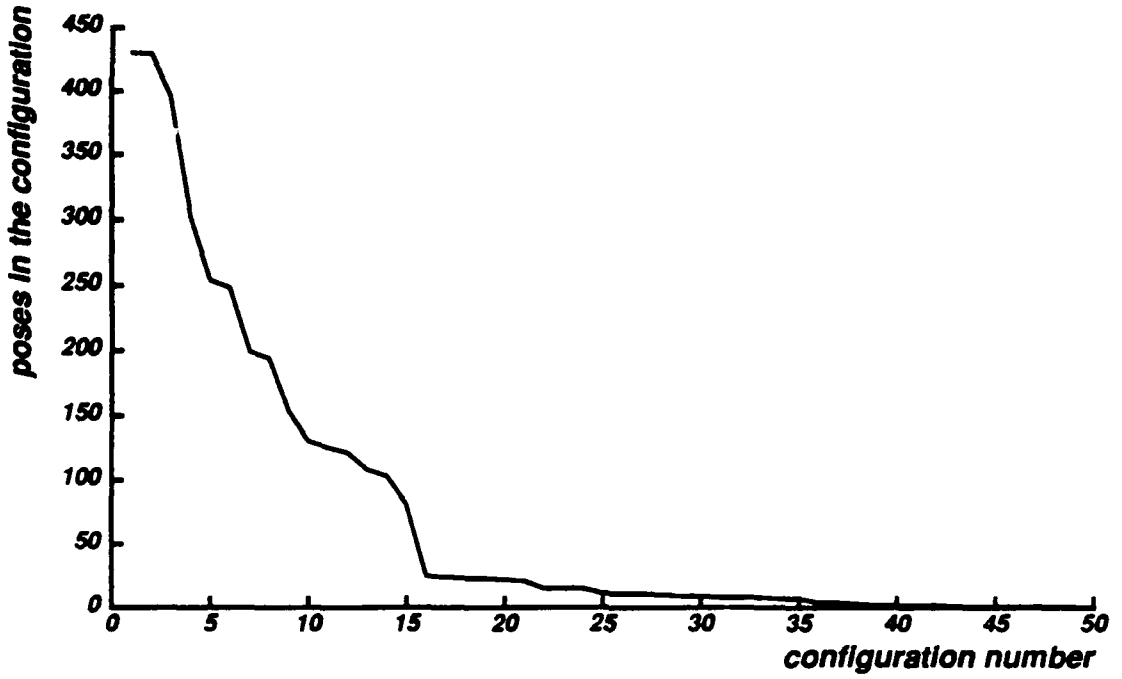


Figure 4.7: *Distribution of pose configurations. The number of poses in each configuration, sorted by size, is shown here.*

Thus, the use of this interpolation approach can be summarized as follows:

1. Sort the memory by configuration.
2. Perform a regression on each configuration to identify the a coefficients.
3. Build a coarse memory (one with relatively large buckets) to map hand shapes to contact configurations.

Note that to perform the regression analysis for obtaining the a coefficients, at least as many equations as coefficients are required. If a particular configuration lacks the required number of entries, additional ones can be obtained by sampling the space more finely in the deficient area.

To demonstrate the interpolation ability of this technique, a number of tests were performed on a sample memory generated using objects grasped by a two fingered, two

jointed hand. The following paragraphs describe these tests and present their findings.

First, an examination of the number and distribution of configurations found in a memory was performed. Figure 4.7 shows the distribution for an object with three vertices. A uniform tessellation of the object position space was performed, with 8000 total positions sampled. Note that the first 15 or so configurations account for the bulk of the sampled grasps. Several of the configurations lack the required minimum number of poses for identifying the regression coefficients. Finer tessellation in those areas would be required.

For each of the configurations that had at least the minimal number of required poses, a regression was performed. The Levenberg-Marquardt algorithm, as implemented by the Minpack [73] library, was used. The algorithm obtained the set of coefficients that accurately recovered the joint angle to pose mapping for the data that was used in the regression. In all cases, a stable set of coefficients that satisfied the minimization could be found. The coefficients, if based on a sufficient sampling of the configuration space, were found to allow a reasonably accurate interpolation between entries. The experiments suggest that the number of pose samples required for obtaining a good estimate of the coefficients varied widely. As one would expect, for hand shapes that covered a large range of positions, a relatively large number of pose samples were required. For smaller ranges of hand shapes, as few as 25 samples were required for obtaining reasonable position estimates. This supports the claim that a second order function is appropriate for mapping joint angles to an object pose.

The following procedure was used to examine the interpolation capabilities in more detail. The memory entries for an exemplar contact configuration were randomly ordered. The first n of these entries, where n is varied from the minimum required samples for interpolation, to the maximum number of entries in the configuration, were used in the regression. The remaining entries from the configuration, the ones not used in the regression, were then predicted using the computed coefficients. The error between the

x			y			θ		
desired	actual	error	desired	actual	error	desired	actual	error
-0.200	-0.188	1.1	0.600	0.626	2.9	0.400	0.380	3.3
-0.300	-0.262	3.5	0.200	0.267	7.4	0.500	0.448	8.6
0.000	-0.148	13.4	0.200	0.585	42.8	0.600	0.366	39.1
0.200	0.190	0.9	0.800	0.773	3.0	0.400	0.400	0.1
-0.500	-0.501	0.1	0.100	0.165	7.3	0.400	0.352	8.1
-0.600	-0.620	1.8	0.300	0.404	11.5	0.300	0.243	9.4
0.100	0.121	2.0	0.900	0.759	15.6	0.300	0.380	13.3
-0.200	-0.195	0.5	0.800	0.627	19.2	0.100	0.202	17.0
0.200	0.172	2.5	0.500	0.727	25.2	0.300	0.173	21.2
0.300	0.303	0.2	0.800	0.798	0.2	0.300	0.304	0.7
-0.400	-0.371	2.6	0.000	0.020	2.3	0.500	0.476	4.1
0.100	0.079	1.9	0.600	0.617	1.9	0.100	0.082	3.0
-0.300	-0.264	3.2	0.200	0.197	0.4	0.400	0.398	0.4
0.300	0.308	0.8	0.600	0.741	15.6	0.300	0.217	13.9
0.400	0.440	3.6	0.600	0.756	17.3	0.300	0.204	16.1
-0.300	-0.278	2.0	0.600	0.539	6.8	0.200	0.241	6.9
0.100	0.082	1.6	0.500	0.637	15.2	0.200	0.122	13.1
0.500	0.504	0.4	0.800	0.813	1.5	0.100	0.128	4.7
0.300	0.317	1.6	0.700	0.671	3.3	0.100	0.111	1.8
0.000	-0.043	3.9	0.500	0.674	19.4	0.400	0.298	17.0
-0.200	-0.172	2.5	0.200	0.076	13.7	0.300	0.365	10.8
0.500	0.478	2.0	0.800	0.875	8.3	0.300	0.257	7.2

Table 4.1: *Interpolated poses. This table shows the desired position parameter, the actual parameter computed by the estimation function, and the percent error of that value.*

predicted and actual values were then found. A percent error was also computed, based on the total range in values for that parameter.

Table 4.1 presents a number of pose location parameters that were computed using the estimation function. Notice that in most cases, the percent error in position is small, usually only a few percent. The error in θ is slightly larger in general, though this is probably because of the larger tessellation that was performed in that dimension.

Two configurations were compared, one that covered a larger range of object positions than the other. The first had 107 entries, and covered the smaller range. Figure 4.8

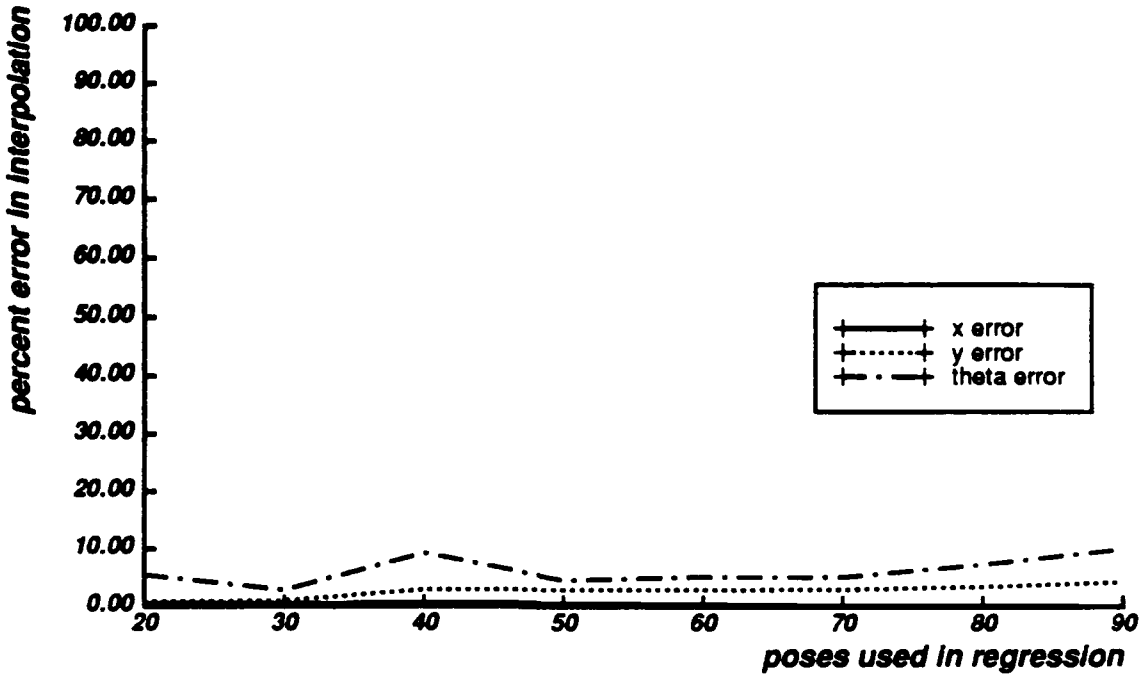


Figure 4.8: *Typical interpolation error. The error in interpolation, for object x, y, and θ, is plotted against the number of configurations used in the regression. This plot is representative of the performance of the method under most tested configurations.*

plots the error in x, y, and θ for a randomly selected position, as additional poses are used in the regression. The performance here is quite good. Figure 4.9 pl · the same parameters for a larger configuration, one with 252 entries. In this case, performance is somewhat degraded.

The stability of the estimation function's coefficients was examined. Figure 4.10 shows a plot of three coefficients against the number of samples used in the regression. Notice that as the number of samples increases, the coefficients approach a constant value, as would be expected. For the trial plotted in the figure, approximately 125 poses were required before a stable set of coefficients were found. This particular configuration covered a rather wide range of hand shapes, and needed a large number of poses to obtain a stable set of coefficients. For configurations that occurred over a smaller range of hand shapes, fewer coefficients were usually required. For some cases, as few as 25 poses

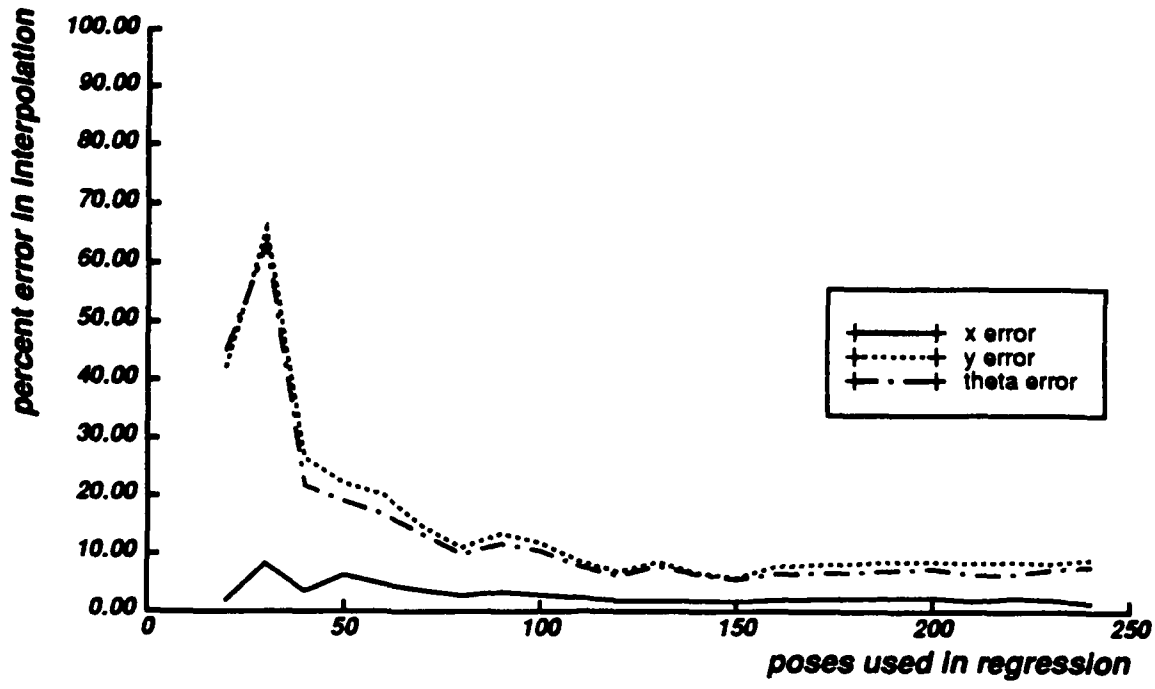


Figure 4.9: *Large interpolation error. The error in interpolation, for object x, y, and θ. is plotted against the number of configurations used in the regression. Initially, the error in predicted position parameters is larger, due to the larger range of positions that are in this configuration.*

produced reasonable results.

4.4.6 Execution Algorithms

In the next sections, two of the potential memory determination strategies are examined in more depth. The first method considered is for tessellated object space with pre-computation of the memory. The second considered is for tessellated hand space with on-demand computation of the memory.

Pre-Computed Tessellated Object Space

Pre-computed tessellated object space is appropriate when the number of object configurations is smaller than the number of hand configurations, and either the number of

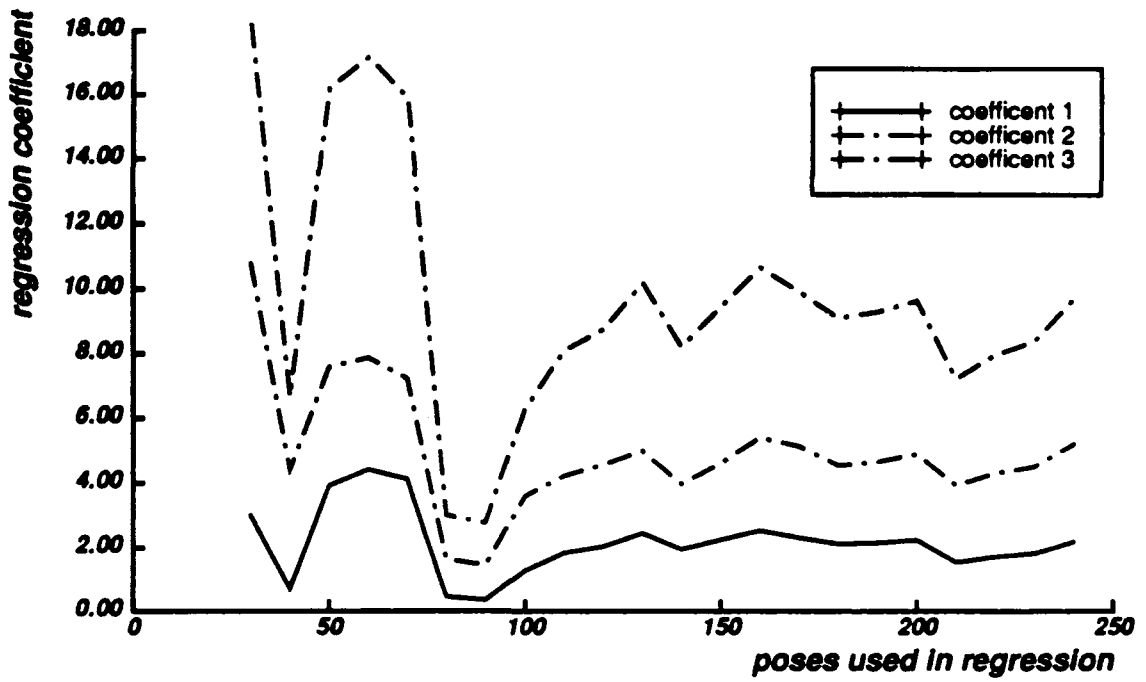


Figure 4.10: *Estimation function coefficients. Three of the function's coefficients are plotted against the number of samples used in the regression.*

hand configurations is small, or a good grasp simulator exists. The algorithm has two stages, the pre-computation stage to fill the memory, and the run-time determination stage.

The determination algorithm, using interpolation, proceeds as follows (see also Figure 4.11):

1. Pre-computation stage:

   (a) Build the determination memory by sampling object configurations and simulating grasps.

2. Determination stage:

   (a) Obtain the determination memory key from the set of hand joint angles.

   (b) Index into the determination memory using the key.

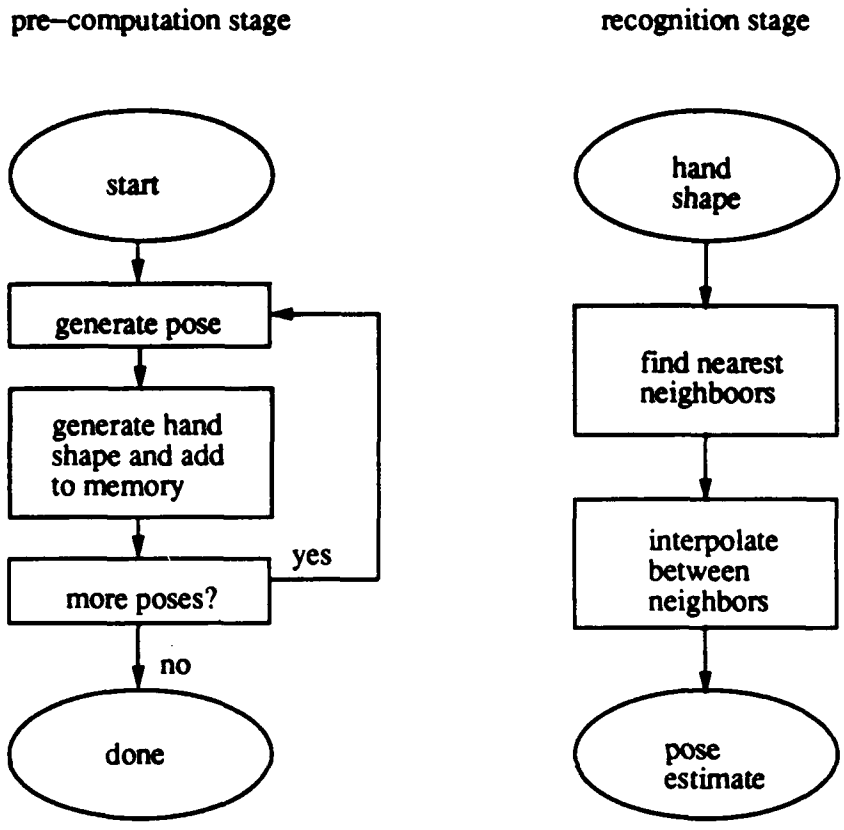


Figure 4.11: *Pre-computed tessellated object space. This figure shows a flowchart for the determination algorithm.*

(c) Order candidates from memory entry based on the closest matches to the actual joint angles. Interpolate object pose.

If interpolation is not to be used, only the best match to the key is extracted, and is used for the pose estimate.

On-Demand Tessellated Hand Space

On-demand tessellated hand space is appropriate when the number of hand configurations is smaller than the number of object configurations, and either the number of object configurations is small, or a good pose finder exists. The algorithm has just one

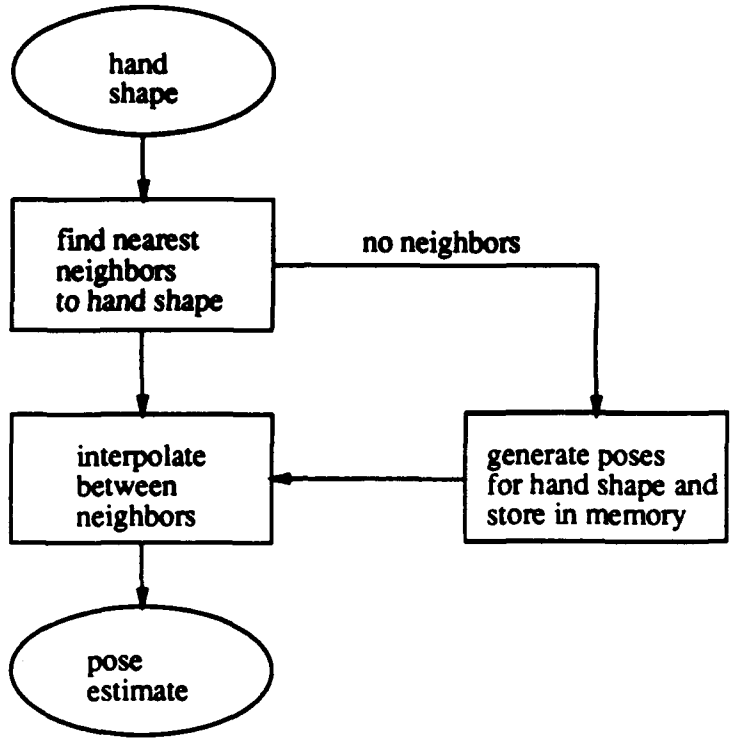


Figure 4.12: *On-demand tessellated hand space. This figure shows a flowchart for the determination algorithm.*

stage, the run-time determination stage.

The determination algorithm, using interpolation, proceeds as follows (see also Figure 4.12):

1. Determination stage:

   (a) Obtain the determination memory key from the hand's joint angles.

   (b) Index into the determination memory using the key.

   (c) If there are no matches, find poses for the hand shape. Add poses to the memory.

   (d) Order candidates from memory entry based on the closest matches to the actual joint angles. Interpolate object pose.

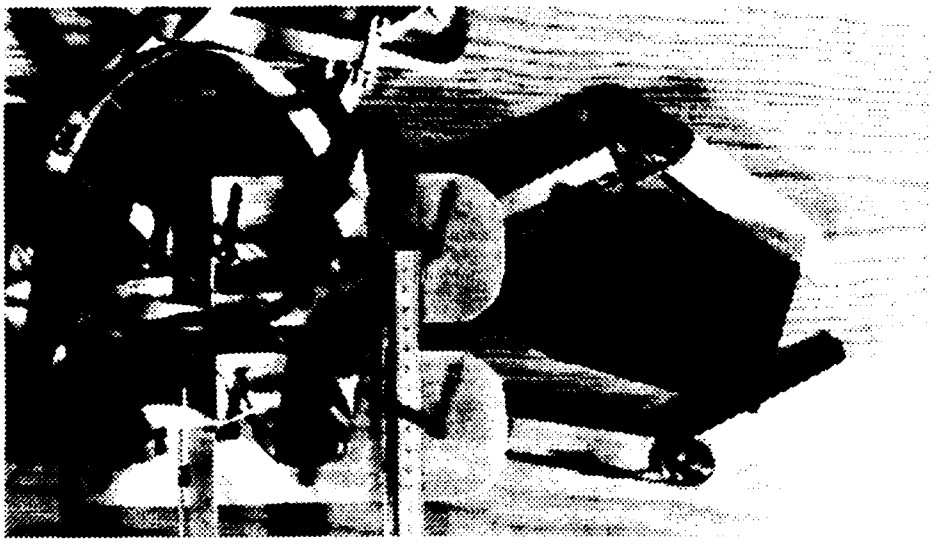


Figure 4.13: *Photograph of the planar hand.*

If interpolation is not to be used, only the best match to the key is extracted, and is used for the pose estimate.

4.5 Experiments

To test the ideas presented in this chapter, both simulations and trials using an actual robotic hand were performed. The simulations examined the characteristics of the determination memories as certain parameters were varied. Experiments on a two-linked, two-fingered planar hand were also performed, which is described in Appendix A. A photograph of the hand is shown in Figure 4.13. Unless otherwise noted, the experiments were conducted using this planar hand with planar objects.

For simplicity, the experiments conducted in this chapter were done using a move-until-contact acquisition strategy. Here, the hand's joints are rolled forward, closing onto an object, until a joint torque limit is exceeded. The joints are closed from proximal to distal. When the most proximal joint can no longer move, the next joint in sequence is then servoed.

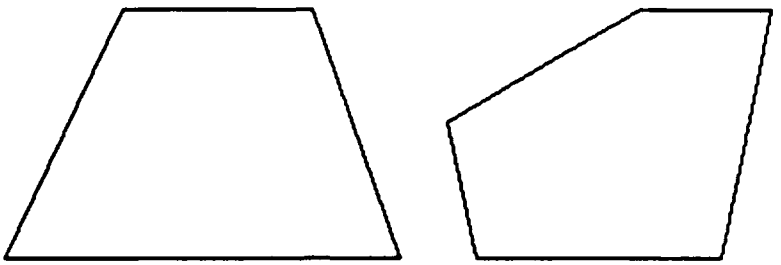


Figure 4.14: *Experimental objects.*

4.5.1 Pose Determination

For these experiments, a library of two objects was used, as shown in Figure 4.14. The pre-computation tessellated object space method was used. By building the memory with two objects, these tests perform both small set recognition and pose determination. For the recognition experiments performed here, different bucket sizes where used, where a determination memory is generated at each bucket size. The memory lookup operation is performed first with the smallest bucket size. If no matches are found, the memory with the next largest sized bucket is then consulted. The process repeats until matches have been found.

Figure 4.15 examines the number of buckets in each memory. The x-axis indicates the relative bucket size. The y-axis indicates the total number of memory entries that were found in that bucket. In the following results, the memories from only the first two buckets are consulted. The larger bucket sizes proved to be unnecessary as they grouped too many entries together.

Trial 1: For this hand shape, there were essentially two poses found, as the two right-most entries are slight variations of the same pose, as shown in Figure 4.16. In this figure, and the subsequent ones, the actual hand shape is drawn in a dark line, while the hand shape from the bucket key is drawn in a light line. The actual object grasped is shown in the right poses. Note that they form a closer fit to the actual hand shape.

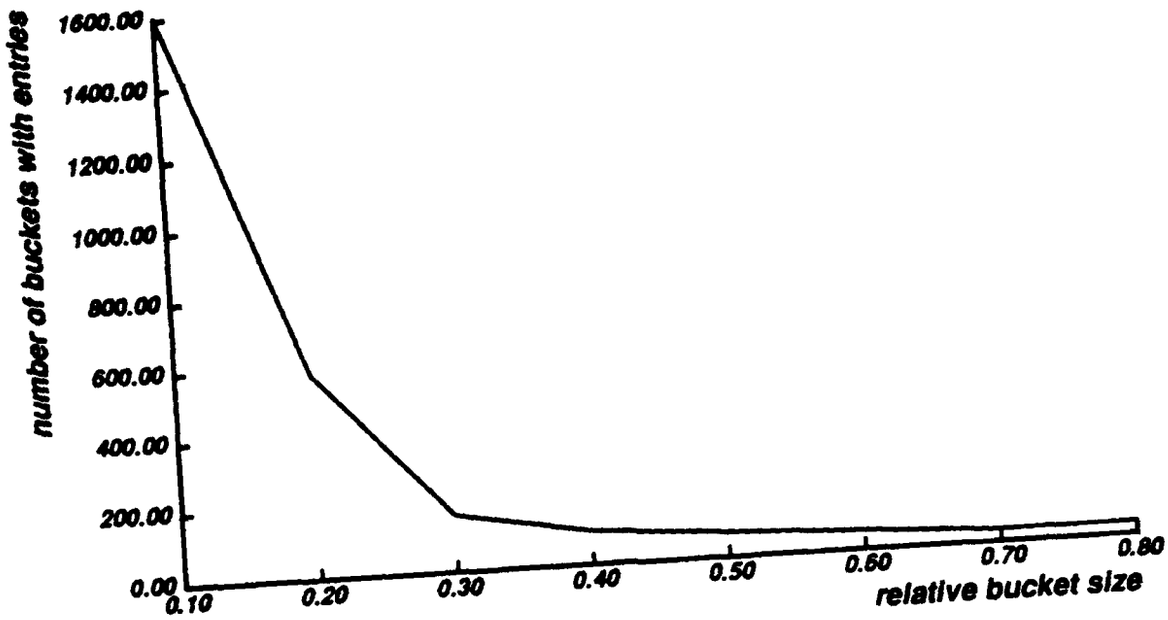


Figure 4.15: *Buckets per determination memory.*

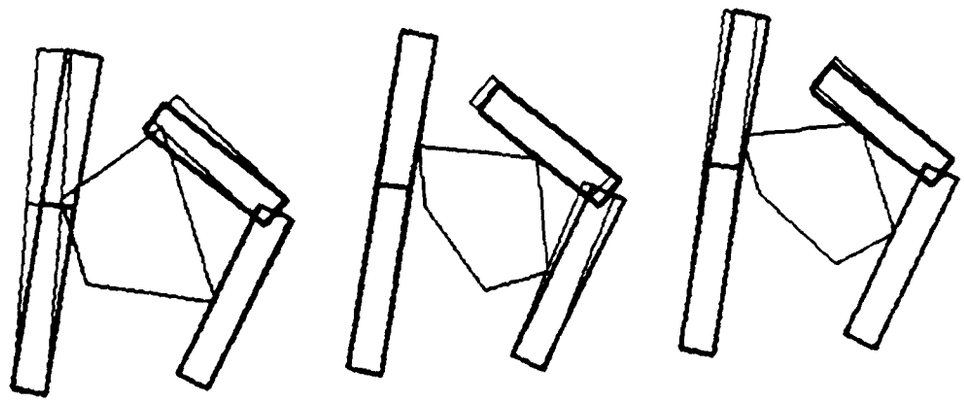


Figure 4.16: *Trial 1: three memory entries.*

All these poses were found from the memory with the finest bucket size.

Trial 2: In this trial, shown in Figure 4.17, the upper two poses were found from the most fine bucket size. The left most of these corresponds to the actual pose of the object. The right pose is from the same family of poses, as well. Note that no poses from the other object in the library were consistent with this hand shape, as is desired. For reference, the lower nine poses are from the memory with the next larger scale. Note that there still are no poses from the other library object in this set.

Trial 3: For the trial shown in Figure 4.18, only one pose was found. This pose corresponds to the grasped object's position.

Trial 4: For the trial shown in Figure 4.19, far more poses were found in the memory entry. The poses shown in the top row of the figure corresponds to the object's position. Note that a large number of poses for the other library object were also found, in two totally distinct configurations. All these poses are from the most fine memory.

Trial 5: For the final trial, shown in Figure 4.20, the top most pose was the only pose that corresponded to the hand shape data, from the most fine determination memory. It correctly corresponds with the pose of the grasped object. The remaining poses were found from the next larger scaled memory entry.

4.5.2 Haptic Information Content of Hand Shape

As more links are added to a hand, its haptic information content should increase. In the limiting case, an infinite jointed hand would totally conform to the shape of the object that it was enclosing, as if it were a rope tied around the object.

One way to investigate the haptic information contained in hand shape is to examine determination memories. An experiment was conducted for this purpose. For a fixed

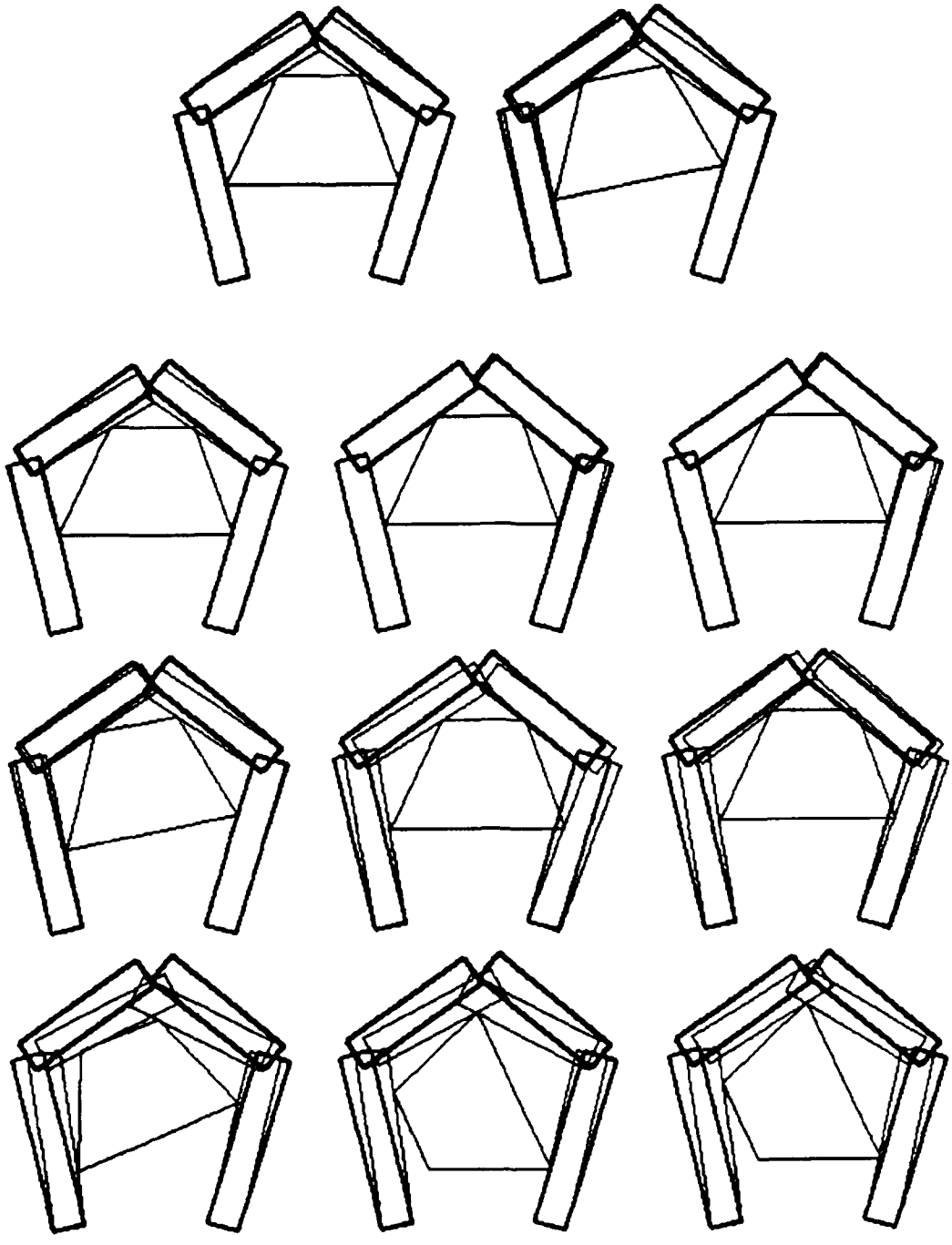


Figure 4.17: *Trial 2: eleven memory entries.*

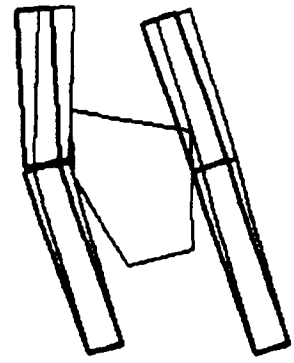


Figure 4.18: *Trial 3: one memory entry.*

object, the number of links on a simulated two fingered hand was varied. The sum of the link lengths was kept constant. For each hand, determination memories were built by tessellating the object configuration space. The recognition power of a hand is defined to be inversely proportional to the ambiguities in the determination memory. An entry is defined to be ambiguous if there are more than a certain low number of poses in an entry bucket.

Figure 4.21 presents the results of this experiment using a bar chart. The x-axis indicates the number of entries in a bucket for that bucket to be considered unambiguous. The y-axis indicates the percent of entries in the entire memory that are unambiguous. Each bar is divided into four categories, which indicate the number of links in each finger. From left to right, two, three, four, and five link fingers are considered. From the plot it can be seen that even in the worst case, with a two link hand, 68 percent of all memory entries are unambiguous. Adding three links dramatically increases the performance making 84 percent of the entries unique. If up to three entries in a bucket are considered unique, 94 percent of the memory entries are acceptable for a four linked finger.

There is an interaction between the number of finger links, the bucket size used to group entries, and the percent of memory entries that are unique. As the entry key's

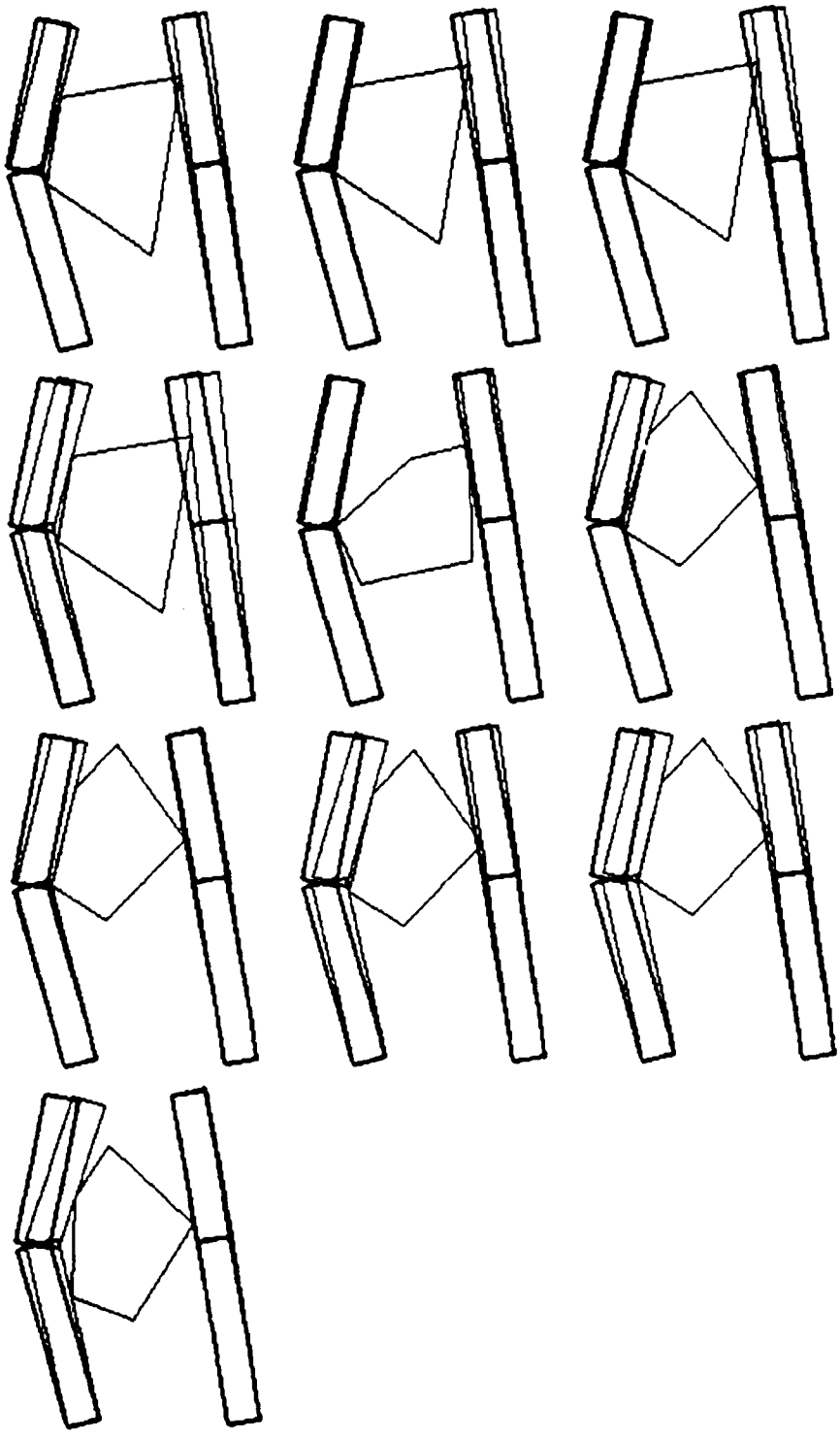


Figure 4.19: *Trial 4: ten memory entries.*

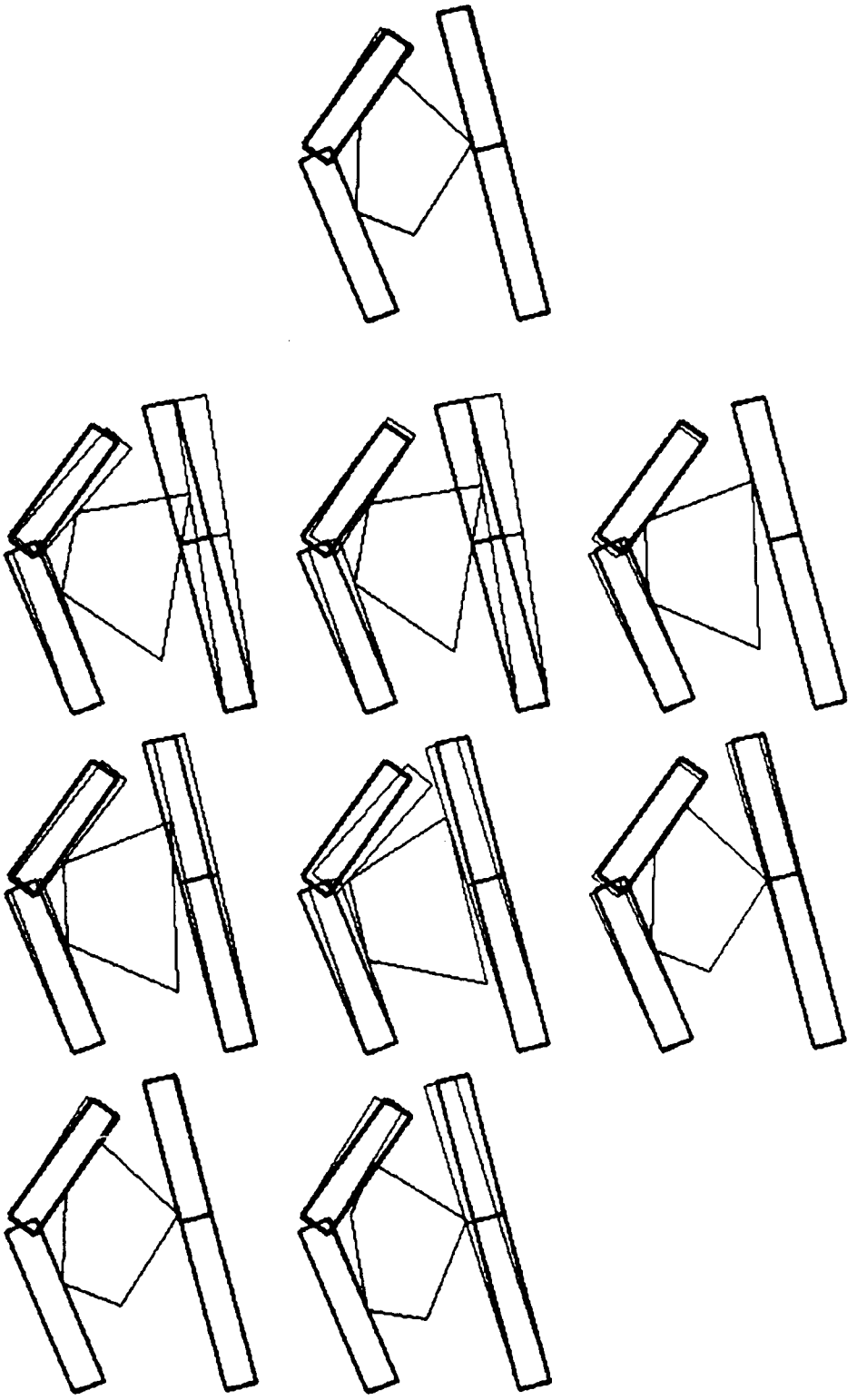


Figure 4.20: *Trial 5: nine memory entries.*

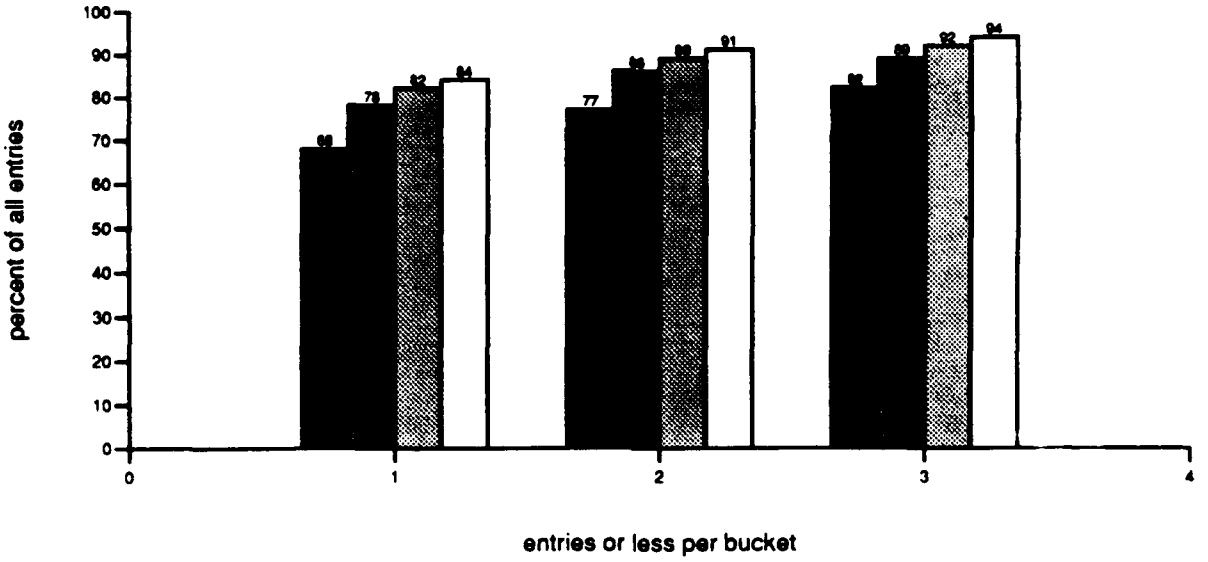


Figure 4.21: *Number of links vs. memory ambiguity. Each cluster contains four bars, one for a two, three, four and five link finger. Each cluster denotes the maximum number of entries in a bucket that is considered to be unique, ranging from one to three. The y-axis of the plot indicates the percent of all entries that are unique.*

bucket size is increased, more entries will be assigned to the same bucket, as there are fewer buckets. The plot in Figure 4.22 shows this effect. The x-axis indicates the relative bucket size. The larger x values indicate larger buckets. The y-axis indicates the percent of memory entries that are unique. The results for two, three, and four link fingers are plotted. Note that as the bucket size increases, fewer entries are unique. Adding more links (which also increase the number of buckets) increases the percent of unique entries for all bucket sizes. It is interesting to note that the reduction in ambiguity as finger links are added is not as dramatic as might be thought.

To consider how sensing might help improve the performance of this determination approach, the following experiment was performed. A memory was generated for a test object. The memory was sorted by number of poses in a bucket, where a bucket holds a set of similar hand shapes. In addition, another memory was built using the same hand and object, but with the additional information that would be provided by sensors that indicate if contact with a link has been made. The sensor information provides a way

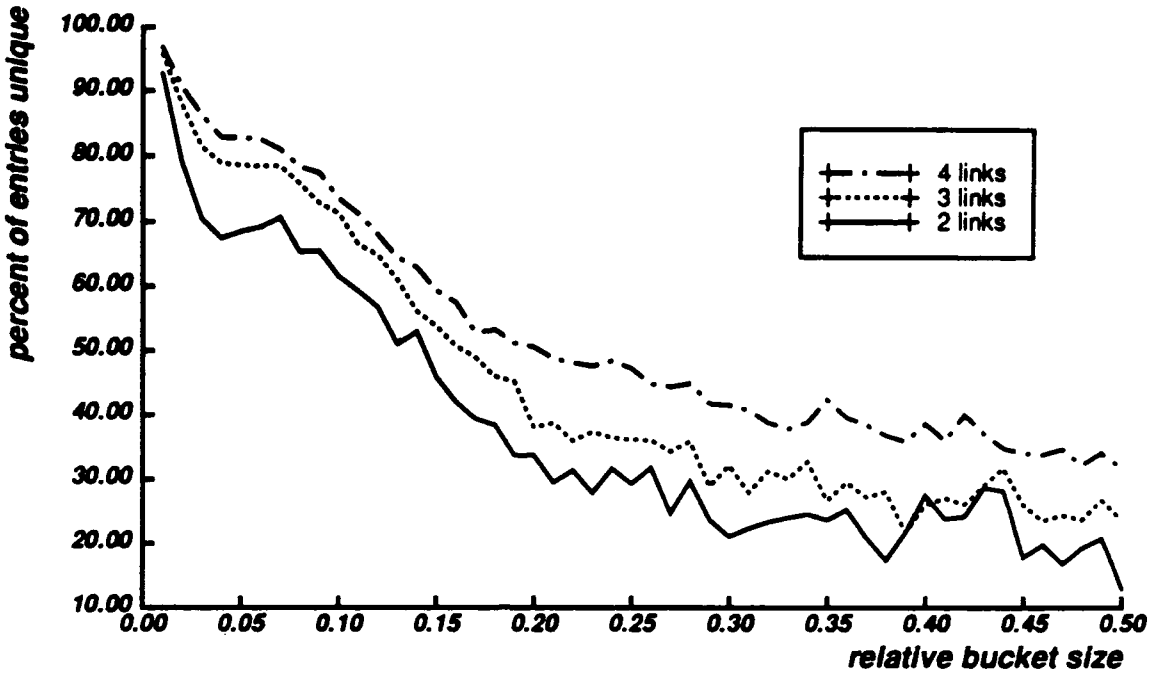


Figure 4.22: *Effect of hand fingers on memory ambiguity. This plot shows that larger number of finger links increases the recognition power of a hand, but not as dramatically as might be expected.*

to disambiguate among multiple poses that map to the same hand shape. If the poses result in a different set of link contacts, they can be distinguished.

In Figure 4.23, the solid line plots the total number of poses that are found in each bucket size. Thus, an xy point indicates the total number of poses (the y value) that are in buckets that have x poses in them. Any bucket that has more than one pose in it is considered ambiguous. The dotted line plots the same numbers for a hand equipped with sensors. To eliminate the effect of related families of similar poses, a family of poses was considered as a single pose when computing these results.

In the plots, notice that the number of poses in the larger buckets drops off significantly when using the contact sensing information. All buckets now have just one or two

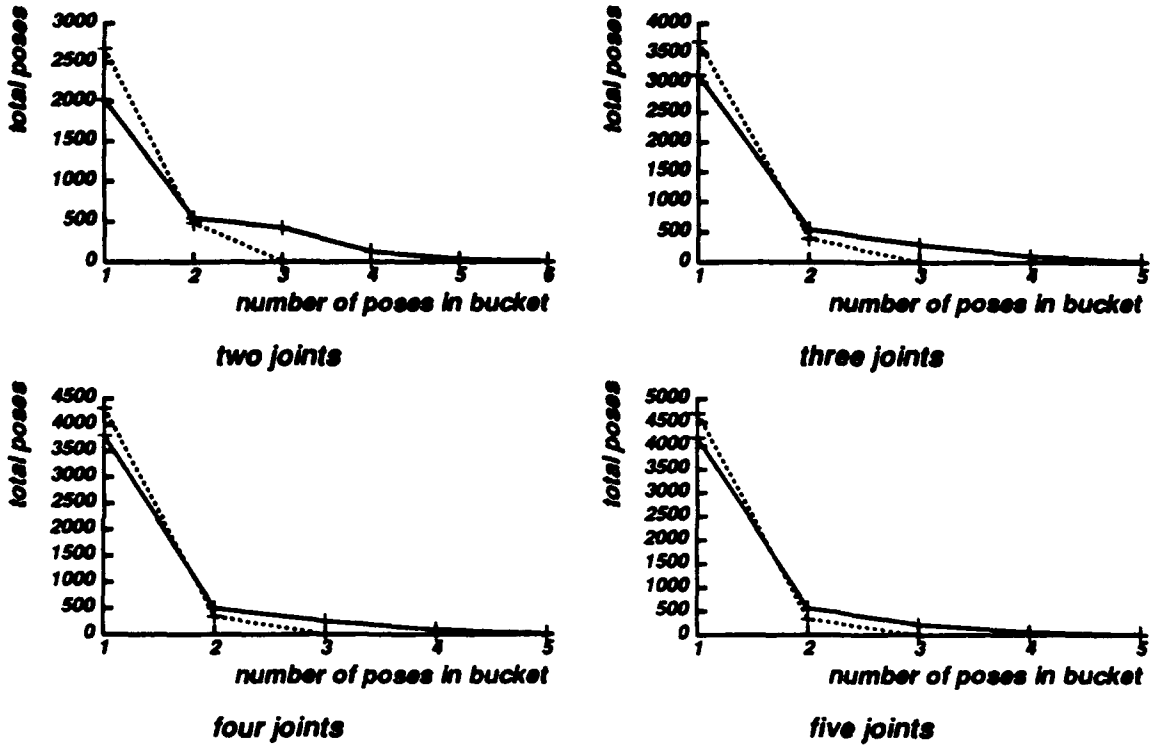


Figure 4.23: *Effect of sensing on memory ambiguity. The numbers plotted with a solid line are for a hand without sensing. The dotted line is for a hand with sensing.*

poses in them. Notice as well that by adding more links to the hand an improvement also results, though not as large. This result suggests that adding contact sensors to a hand improves its recognition power more than would adding a small number of extra finger links.

4.6 Summary and Discussion

This chapter described a memory-based method for localizing objects grasped by a hand. Various approaches for organizing, filling, and using the determination memory were explored. Experiments were performed using both simulations and a planar hand with two fingers, each with two links.

The results show that memory-based determination can be used to localize an object's pose. The use of the grasp acquisition strategy constraint makes the memory size manageable. Memory can be further compacted by using regression-based curve fitting. Regression also provides a way to interpolate between entries to obtain a more precise pose estimate.

For problems in full three dimensions, memories are still useful, though certain factors must be considered. Tessellations of the object space may be practical only under certain restricted conditions. For the full three dimensions, six parameters must be varied. For all but very coarse spacing, it would be too time consuming to perform such a tessellation. However, if objects are resting on a stable face on a table, and if the hand approaches the object from a fixed direction, such tessellations may be possible. In this case, n tessellations in two dimensions would be required, where n is the number of object faces. For tessellated hand configurations, three dimensions can be handled as long as the grasp acquisition strategy restricts the dimensions of the hand space adequately.

Sensor-Based Pose Refinement

Chapter 5

5.1 Introduction

When manipulating a grasped object, especially with a robot hand, it is helpful to have an estimate of the object's orientation within the grasp. The object's orientation can be extracted from knowledge of the surface normals at the various points of contact with the object, but these surface normals must first be transformed into a common coordinate system. Successful execution of these transformations requires a prohibitive amount of accuracy in calibration of the arm and hand. This chapter presents an on-line method to improve these transformations in spite of calibration errors. The method requires collecting contact force readings as the object is manipulated, and computing transform corrections that minimize the variation in the sum of the contact forces. Both experimental and simulated results are presented, and the implications of the results are discussed.

5.1.1 Relevance of this Problem

When an object is grasped by a robot hand, the eventual position and orientation, or pose, of the object is difficult to accurately predict. When performing even the most rudimentary manipulation of the object (for example, placing it somewhere else), it is helpful to have some additional knowledge of the object's pose relative to a fixed coordinate frame.

Previous chapters of this report developed techniques for coarse pose estimation relative to the reference frame of the hand. In some cases, more precise estimates are required. In addition, since the previous techniques used just hand shape and grasping strategy information, solution ambiguities could occur. Additional sensor information can be used to overcome these problems.

Sensor-based refinement techniques must combine local sensor readings into a global pose estimate. This requires accurate transforms from local coordinate frames to a global coordinate frame. This chapter shows that the calibration and sensing requirements for obtaining accurate global data are severe. In fact, simple operations such as adding the contact forces together to find the object's weight gave meaningless results. Techniques for self-calibration are presented in this chapter, as they can reduce this problem.

5.1.2 Source of Information

There are various sources of information available for finding the pose of an object relative to a fixed world coordinate frame. One source of global information is the set of joint position sensors. The locations of the intended contact points can be calculated using the robot's forward kinematics. For fingertip grasps, for example, the contact points will be at the locations of the fingertips themselves. This information aids in determining the pose of the object, but only within the limits of calibration and model errors.

Another source of information is force and surface normal measurements at the points

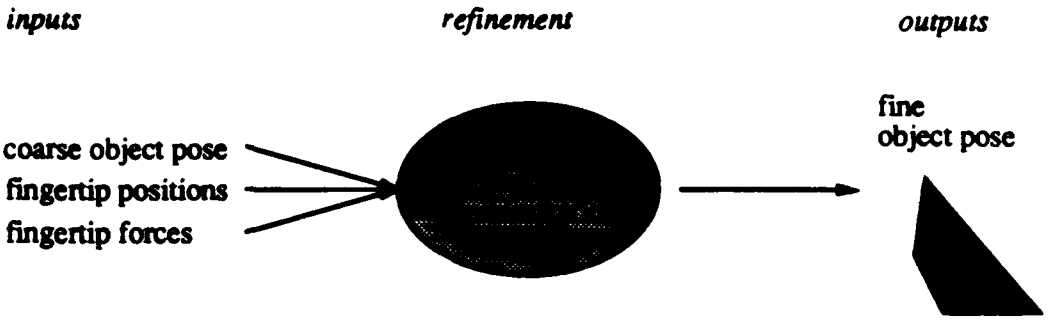


Figure 5.1: *Overview of pose refinement.*

of contact. Fingertip sensors, such as those designed by Brock and Chiu [15], can provide this type of information. Surface normals, in particular, can be fit to an object model to solve for the object's orientation. Since the surface normal measurements are local measurements, however, they must be transformed into a fixed frame before they can be of any use. One way to do this is simply to use the robot's inverse kinematics. Of course, this is susceptible to the same calibration and model errors as the first method.

In principle, the object's orientation estimate can be improved by assuming that there are calibration errors, and by attempting to solve for corrections to the transforms from the contact points to a fixed coordinate frame. This can be done by taking a number of force readings at different object orientations and finding the correction transforms that best fit the requirement that the sum of the contact forces must equal the object weight at every object orientation. Here, contact forces are used as an independent information source to augment the information we have on the configuration of the robot. Figure 5.1 diagrams the process used to refine an object's pose.

Calibration problems are compounded by the object grasping forces. When the fingers constrain an object, they exert an internal force. This force can be quite large, depending on the grip's firmness. The *signal* being extracted from the fingertip sensors is the total external force applied to the object due to gravity. The *noise* that must be

overcome is the internal grasping force. It is typical for an object to be grasped with an internal force several times that of its weight. This problem can only be solved by having a good calibration.

From another standpoint, this work provides a method for continuously calibrating the fingertip sensors. As an object grasped by a hand is moved by an arm, continuous sensor readings can be obtained and used to refine the system's estimate of the object and fingertip orientations. The relatively small amount of computation required is suitable for real-time applications.

5.1.3 Using World Invariants

The first part of the pose refinement method uses the constraint that the sum of the internal grasping forces must equal the object weight, or simply a constant vector oriented in the direction of gravity. Sensor readings are taken in the fingertip or sensor frame. Errors are introduced when these readings are transformed to the common world frame. The method then attempts to find a correction to each fingertip's orientation to make the constraint on the sum of the forces hold true, after the tip forces are transformed to a common coordinate frame.

5.1.4 Chapter Overview

The following sections will show how contact force and surface normal information can aid in determining the orientation of a grasped object. In particular, multiple samples of contact force information are used to help correct for robot calibration and model errors. Section 5.2 provides an overview of previous work. Section 5.3 outlines the assumptions required for the solution given. The approach is discussed in Section 5.4. Section 5.4.1 covers the process of finding an object's orientation with perfect kinematics and sensors, using a set of three linearly independent surface normal measurements. Section 5.4.2 shows how this estimate can be improved by taking force readings with

the object in various orientations and grounding the resultant of the forces to the object weight. Section 5.4.3 shows an alternative approach for refinement based on a small-angle approximation. Section 5.5 discusses the experimental setup and their results. Section 5.6 develops a simulator used for predicting fingertip sensor readings. The simulations are used to examine the performance of the refinement algorithms, and to better understand the observed experimental results. Finally, Section 5.7 presents conclusions.

5.2 Related Work

Related work falls into the categories of haptic object recognition and kinematic calibration. Haptic object recognition is the problem of using the sensors on a robot hand to recognize an object. This problem is often examined from a model-based feature matching framework, such as in Gaston and Lozano-Pérez [38] and Ellis [31]. Object properties such as face normal directions and contact point distances are measured and matched against a model. In contrast, Lederman and Klatzky [67] examine how hand *motions* can be used to recognize objects. Allen and Bajcsy [3] and Allen [2] propose a method for building surface maps using tactile sensors and vision. The surface features can be matched against an object database to identify the object and its pose. These works make no particular mention of the issue of calibration of the hand's kinematics.

Hollerbach [50] provides a recent review of the field of kinematic calibration. Calibration of hands has been studied as a problem of closed-linked kinematic chains by Everett and Lin [32], and Bennett and Hollerbach [10, 9].

This chapter contributes to the work mentioned above by presenting an *active* calibration process that can be used while the hand is performing its normal manipulations. Multiple sensor readings obtained in the course of the motion are used to refine the sensor orientation estimates. This is made possible because we only need to correct *fingertip orientation* in order to extract *object orientation.* Locally correct estimates of

object orientation are acceptable, as long as they are kept up to date. This means that complete calibration of the configuration of the arm and hand is not required.

5.3 Assumptions

The following assumptions are made in this chapter:

- Contact forces and estimated surface normals can be obtained at all points of contact with the object.
- Known *changes* in object orientation can be generated (this may correspond to having accurate arm kinematics and inaccurate hand kinematics).
- The grasped object is polyhedral, and the faces containing each of the contact points are known.
- The grasped object is assumed to be in static equilibrium.
- The error in the contact force and surface normal measurements (the sensor readings) is negligible in comparison to the error in the position and orientation of the sensor frame.

5.4 Approach

The pose refinement process is described in two stages. First, a method is presented for refining fingertip contact normals and finding the weight of the grasped object. This method relies only on fingertip sensor data. Next, the added constraints provided by the object's model and the fingertip to object face assignments is used to refine the object's orientation.

Several reference frames that are used in the subsequent sections are first defined:

t_i fingertip i frame

w_i world frame seen by fingertip i

w world frame

The fingertip world frames are obtained from the robot's forward kinematics and the set of joint position sensors. They are inaccurate due to calibration errors. Essentially, the method described here will find a correction to take w_i to w for each fingertip.

5.4.1 Recovering an Object's Pose

If the transforms obtained from the inverse kinematics of the robot are accurate, finding the object orientation is easy. To perform an unambiguous fit, contact information must be available on three surfaces of the object with linearly independent normals. The object orientation desired is the best rotation of the modeled object to bring the surface normals M_i^w into alignment with the measured surface normals N_i^w. The measured surface normals N_i^w are transformed from the local sensor frames to the world frame by

$$N_i^w = T_{t_i}^w N_i^{t_i}. \tag{5.1}$$

The object model normals must then be aligned with the set of world coordinate normals, N_i^w. One error function that could be used to perform the alignment is given by

$$E = \sum_{i=1}^{3} (\Delta\theta_i)^2, \tag{5.2}$$

where $\Delta\theta_i$ is the angle between the measured and model normals in the proposed object orientation. The object orientation that minimizes the error term is the most consistent guess available for the given normal measurements.

5.4.2 Refining Inaccurate Contact Normals

In reality, the inverse transforms from the contact sensors local frame to a world frame are not particularly accurate. For a setup using a Salisbury [90] hand and a Puma arm (see Appendix A), the contact forces obtained are so inaccurate that a simple experiment to weigh an object gives meaningless results. Thus, before performing the procedure described in the previous section, corrections to the fingertip transforms $T_{t_i}^w$ must be

found. This section details a method to find these transforms by combining multiple sensor readings using the direction of gravity (or the direction of the object weight) as an invariant.

If the inaccurate world frame for tip i is denoted w_i, then an accurate surface normal N_i^w is found from

$$N_i^w = T_{w_i}^w T_{t_i}^{w_i} N_i^{t_i}, \tag{5.3}$$

where $T_{w_i}^w$ is the correction transform that must be found. The constraint used here is that the sum of the fingertip forces must equal the object weight:

$$\sum_{i=1}^{3} F_i^w = m\vec{g}. \tag{5.4}$$

And of course, the contact forces are transformed to the world frame in the same manner as the surface normals:

$$F_i^w = T_{w_i}^w T_{t_i}^{w_i} F_i^{t_i}. \tag{5.5}$$

If we put the gravity vector along the z-axis, we get:

$$\begin{aligned} \sum_{i=1}^{3} F_{ix}^w &= 0 \\ \sum_{i=1}^{3} F_{iy}^w &= 0 \\ \sum_{i=1}^{3} F_{iz}^w &= mg. \end{aligned} \tag{5.6}$$

The correction transforms $T_{w_i}^w$ must be such that they satisfy Equations 5.5 and 5.6. If the fingertip sensors are sampled in multiple object orientations, an error function can be defined as:

$$E_s^2 = \sum_{i=1}^{3} \left\{ (F_{ixs}^w)^2 + \left(F_{iys}^w\right)^2 + (F_{izs}^w - mg)^2 \right\}. \tag{5.7}$$

If p force samples are collected, the values of $T_{w_i}^w$ that best satisfy

$$\sum_{s=1}^{p} E_s^2 \approx 0, \tag{5.8}$$

give refined sensor to world transforms that can be used to update the contact force and surface normal measurements.

One added problem that must be considered is the weight of the tip itself, which is a significant 0.16N. The force output F will only be accurate when the tip is in its calibration orientation. To correct for this, the tip is calibrated with its z-axis parallel to the gravity vector in world coordinates. A good assumption is that in this configuration, the tip weight acts directly through its origin. Thus, the force that was erroneously subtracted out during calibration can be restored by adding the tip weight to the z-component of all calibrated sensor force readings. These can then be corrected for the effect of the weight in the current configuration by projecting the weight into the current tip frame and subtracting this from the result. The accuracy of this correction, of course, depends on the accuracy of the transform from the world coordinate system to that of the tip. More formally, if $F_i^{t_i\prime}$ are the raw tip force readings, the corrected readings, $F_i^{t_i}$, can be obtained from

$$F_i^{t_i} = F_i^{t_i\prime} - T_w^{t_i} W_i^w, \tag{5.9}$$

where W_i^w is the weight vector of fingertip i, in world coordinates. Thus, Equation 5.5 can be written in terms of the raw forces as

$$F_i^w = T_{w_i}^w T_{t_i}^{w_i} \left(F_i^{t_i\prime} - T_w^{t_i} W_i^w \right). \tag{5.10}$$

The error term in Equation 5.7 is computed using the forces obtained from this equation.

The three correction transforms have a total of nine unknown rotation parameters, but the optimization method can only be used to solve for eight of these. The method is useful for making the tip world coordinate frames internally consistent, and for aligning the z-axes of these frames with the actual world z-axis (or with the direction of gravity), but it cannot align the tip world x and y axes with those of the actual world coordinate system. While the z-axis is in the direction of gravity, there is no natural phenomenon to distinguish x from y. Because of this, a zyz Euler angle representation for the correction

rotations is used (Craig [25]),

$$\begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \tag{5.11}$$

and the last rotation about the z-axis in the correction vector of one of the tips is arbitrarily set to zero. After a solution is found, a rotation about z is applied to all of the tip rotation corrections to minimize the sum of the squares of the angles of correction from the initial guess, or the guess obtained from the robot inverse kinematics. The initial guess may not be very good, but it is the best independent estimate available. Note that the magnitude of the object weight, m, can be supplied as an additional parameter in the minimization if it is not known.

Once the correction transforms have been found, they can be used in Equation 5.3 to transform the measured surface normals, which can then be used to find the object orientation as shown in the previous section.

5.4.3 Refinement Using Small Angle Approximation

One problem with using the minimization numerical technique for finding the tip orientation corrections is its susceptibility to local minima. As with any minimization, if the function being solved contains local minima, false solutions may be obtained. Given a good initial guess, local minima will hopefully be avoided. However, non-numerical techniques are certainly better. This section develops an alternative approach based on a small angle approximations of the rotation matrix that represents a tip correction. This approach has the advantage of linearizing the problem, giving a system of equations that can be solved using a matrix inversion.

The *zyz* Euler angle representation for a rotation given by Equation 5.11 can be

linearized (Paul [81]) to

$$\begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix}. \tag{5.12}$$

Using this linearized form of a rotation, and from Equations 5.5 and 5.6,

$$\sum_{i=1}^{3} \begin{bmatrix} 1 & -\alpha_i & \beta_i \\ \alpha_i & 1 & -\gamma_i \\ -\beta_i & \gamma_i & 1 \end{bmatrix} \begin{bmatrix} f_{xi}^{w_i} \\ f_{yi}^{w_i} \\ f_{zi}^{w_i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}. \tag{5.13}$$

In this form, the desired correction angles are obtained by solving for the $\alpha\beta\gamma$ Euler angles. The forces, f are obtained from the sensors, and transformed to the nominal world coordinate system using the modeled kinematics. Rearranging Equation 5.13 into the form

$$AX = B, \tag{5.14}$$

and eliminating the equation for the final z rotation (which cannot be solved for), gives

$$\begin{bmatrix} 0 & f_{1z}^1 & -f_{1y}^1 & 0 & f_{2z}^1 & -f_{2y}^1 & 0 & f_{3z}^1 \\ -f_{1z}^1 & 0 & f_{1x}^1 & -f_{2z}^1 & 0 & f_{2x}^1 & -f_{3z}^1 & 0 \\ f_{1y}^1 & -f_{1x}^1 & 0 & f_{2y}^1 & -f_{2x}^1 & 0 & f_{3y}^1 & -f_{3x}^1 \\ 0 & f_{1z}^2 & -f_{1y}^2 & 0 & f_{2z}^2 & -f_{2y}^2 & 0 & f_{3z}^2 \\ -f_{1z}^2 & 0 & f_{1x}^2 & -f_{2z}^2 & 0 & f_{2x}^2 & -f_{3z}^2 & 0 \\ f_{1y}^2 & -f_{1x}^2 & 0 & f_{2y}^2 & -f_{2x}^2 & 0 & f_{3y}^2 & -f_{3x}^2 \\ 0 & f_{1z}^3 & -f_{1y}^3 & 0 & f_{2z}^3 & -f_{2y}^3 & 0 & f_{3z}^3 \\ -f_{1z}^3 & 0 & f_{1x}^3 & -f_{2z}^3 & 0 & f_{2x}^3 & -f_{3z}^3 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \\ \alpha_2 \\ \beta_2 \\ \gamma_2 \\ \alpha_3 \\ \beta_3 \end{bmatrix} =$$

$$\begin{bmatrix} -f_{1x}^1 - f_{2x}^1 - f_{3x}^1 \\ -f_{1y}^1 - f_{2y}^1 - f_{3y}^1 \\ mg - f_{1z}^1 - f_{2z}^1 - f_{3z}^1 \\ -f_{1x}^2 - f_{2x}^2 - f_{3x}^2 \\ -f_{1y}^2 - f_{2y}^2 - f_{3y}^2 \\ mg - f_{1z}^2 - f_{2z}^2 - f_{3z}^2 \\ -f_{1x}^3 - f_{2x}^3 - f_{3x}^3 \\ -f_{1y}^3 - f_{2y}^3 - f_{3y}^3 \end{bmatrix} . \quad (5.15)$$

All force readings, f_i^s, are assumed to be in frame w_i. The superscript s refers to the sample number. As before, γ_3 cannot be found. Equation 5.14 has eight equations and eight unknown, and thus A^{-1} is easily obtained. Additional equations can be obtained by using more than the required minimum of three samples. In this case, A^{-1} could be obtained by a pseudo-inverse. *If the object weight was unknown, Equation 5.15 could* be augmented with the equation from the third sample that was omitted.

5.5 Experiments and Results

These experiments test the ideas presented in the previous sections. As outlined above, the direction of gravity is used as an invariant in finding sensor frame correction transforms. The first experiment attempts to weigh objects. While measuring an object's weight is not a particularly interesting experiment in itself, it provides an easy test of the refinement technique. It is much easier to verify an object's weight than its global orientation. The next experiment examines the refinement of contact normals. Again, to avoid the issue of global workspace calibration, objects with parallel faces are grasped, where the fingertip normals are known to oppose each other. Before describing the experiments, the setup and procedure are detailed.

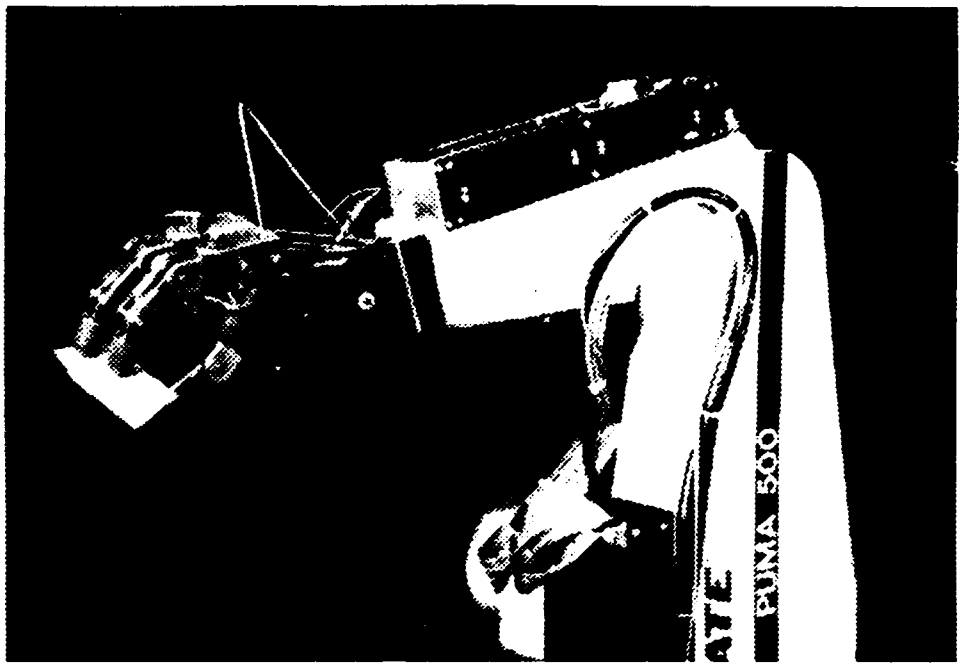


Figure 5.2: *Photograph of the Puma arm with Salisbury Hand.*

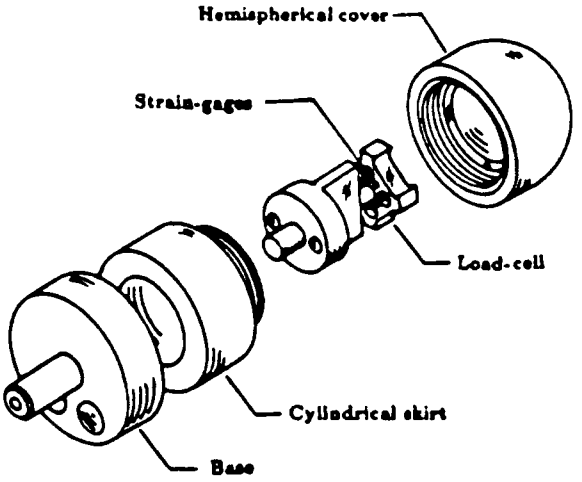


Figure 5.3: *Diagram of the force sensing fingertip.*

5.5.1 Experimental Setup

The experiments described in the next section were conducted using a Salisbury [90] hand mounted on a Puma 500 arm. A photograph of the robot is shown in Figure 5.2. In this setup, contact forces and surface normals are obtained from Brock and Chiu's [15] force sensing fingertip sensors (Figure 5.3). Each sensor has a six axis load-cell, built with eight strain gauges arranged in a Maltese cross configuration. The gauges are paired off with each other, one on each side of the beams that form the cross. The sensor surface is polished aluminum. See Appendix A for a detailed description of the robot and its computational architecture.

The sensors are calibrated using a specially designed apparatus that can apply forces and torques in known directions. The calibration process involves probing the sensor and recording the strain gauge outputs. This data gives a forward calibration matrix. A Morse-Penrose pseudo inverse is computed to obtain a conversion matrix from sensor readings to force values. If C is the experimentally determined calibration data, and S is the 8×1 strain gauge reading vector, then

$$F = C^{-1}S. \tag{5.16}$$

This approach, however, does not take into account the effect of the weight of the fingertip on its own sensor readings. As previously discussed, the orientation of the tip must be known, and an appropriate weight correction must be applied.

Most of the error in estimating the position and orientation of the sensor frames comes from the hand. The Salisbury hand lacks both joint angle sensors on the finger joints and encoder absolute zero marks. Instead, motor positions are used to estimate joint positions. Due to compliance in the tendon system, this estimate is not very accurate. To further compound the problem, the hand must be manually zeroed at startup.

Experimentally, it was found that refinement using the small angle approximation (Section 5.4.3) did not work well. This can be explained by the rather large correction angles that were found to occur. This would cause the small angle assumption to

fail, giving erroneous results. Because of this, the minimization method for finding the corrections was used instead.

The correction transforms were obtained using the Minpack [73] package. This package of Fortran subroutines is used to solve numerical minimizations of systems of equations. In particular, the LMDIF subroutine was used, which implements a modification of the Levenberg-Marquardt nonlinear least squares algorithm. The minimization solved by LMDIF is given by

$$\min\left\{\sum_{i=1}^{p} f_i(x)^2 : x \varepsilon R^n\right\} \tag{5.17}$$

where $f_i(x)$ is obtained from Equation 5.7, and p is the number of sensor samples. LMDIF computes the required Jacobian matrix using a forward-difference approximation.

5.5.2 Weighing an Object

This section shows the results from a few representative attempts to weigh an object. A more general discussion of the results is then presented in Section 5.7 below.

Figure 5.4 shows the measured force sums for all samples of one run along the world x, y, and z-axes. These sums are shown *before* the correction transforms have been applied. The object's actual weight, as shown in the figure, is 1.56N. Note that all the trials in this section used 15 sensor samples for the minimization. As will be discussed in the next section, this turned out to be inadequate. A larger number of samples should have worked much better.

Figure 5.5 shows force sums from the same run *after* the correction transforms have been applied. The object weight was supplied as a variable (with a guess of 1.0N). Note that the force sums in the x and y directions are now nearly zero, as they should be, and the force in the z-direction is nearer to the actual object weight.

Table 5.1 shows the rotation correction Euler angles (in radians) returned from three runs executed at different points in the workspace. The object weights returned from

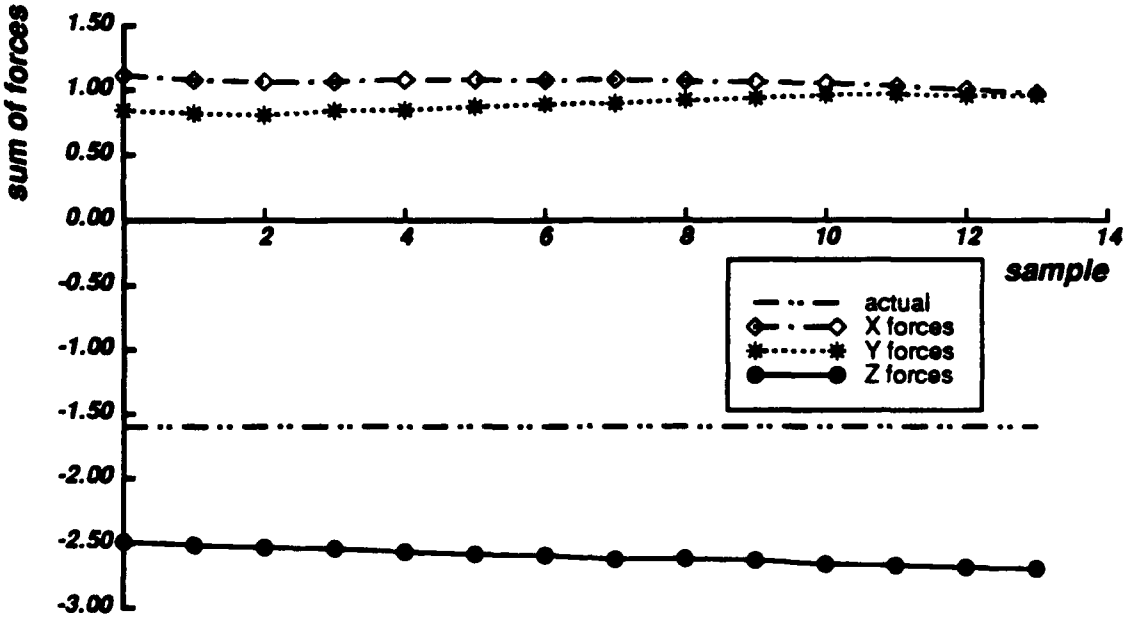


Figure 5.4: *This plot shows the sum of the x, y, and z forces, before the tip orientation corrections have been applied.*

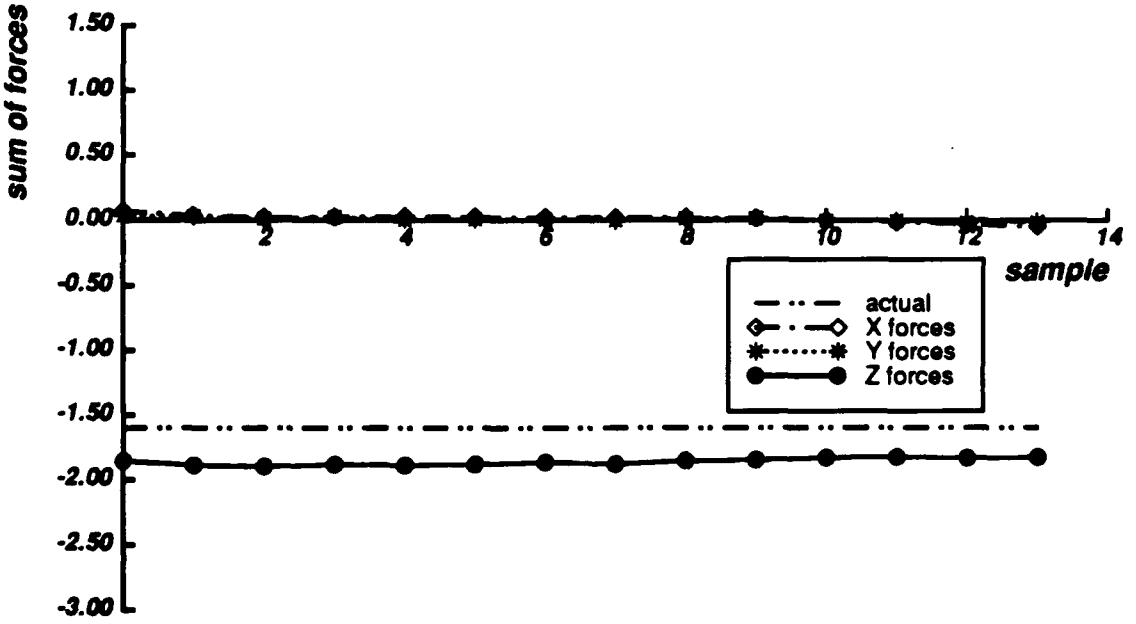


Figure 5.5: *This plot shows the sum of the x, y, and z forces, after the tip orientation corrections have been applied.*

θ	Tip 1	Tip 2	Tip 3
α	-0.067	-0.074	0.187
β	0.105	0.012	0.355
γ	0.312	0.218	0.019

θ	Tip 1	Tip 2	Tip 3
α	0.267	0.694	0.764
β	-0.221	-0.121	0.321
γ	0.138	-0.441	-0.099

θ	Tip 1	Tip 2	Tip 3
α	-0.028	-0.219	0.061
β	-0.428	-0.201	0.284
γ	-0.129	-0.669	-0.098

Table 5.1: *Tip rotation corrections. The rotation corrections in radians, for each tip, for three trials at different points in the workspace are shown.*

Weight	Optimized	Average
1	1.50	1.86
2	2.02	2.18
3	2.17	2.58

Table 5.2: *Optimized and average object weights. The optimized and average object weights (in Newtons) obtained for the same three trials are shown. The actual object weight was 1.56N.*

the same three runs are shown in Table 5.2. The optimized weight is the weight returned by the minimization process, and the average weight is the average over all samples of the magnitude of the corrected force sum. These runs were also done with an initial guess at object weight of 1.0N. The actual object weight is 1.56N. The error terms for the three runs, by sample, are shown in Figure 5.6.

Figure 5.7 shows the effect of supplying different guesses of the object weight on the optimized weight returned. The actual weight of the object grasped in these trials is

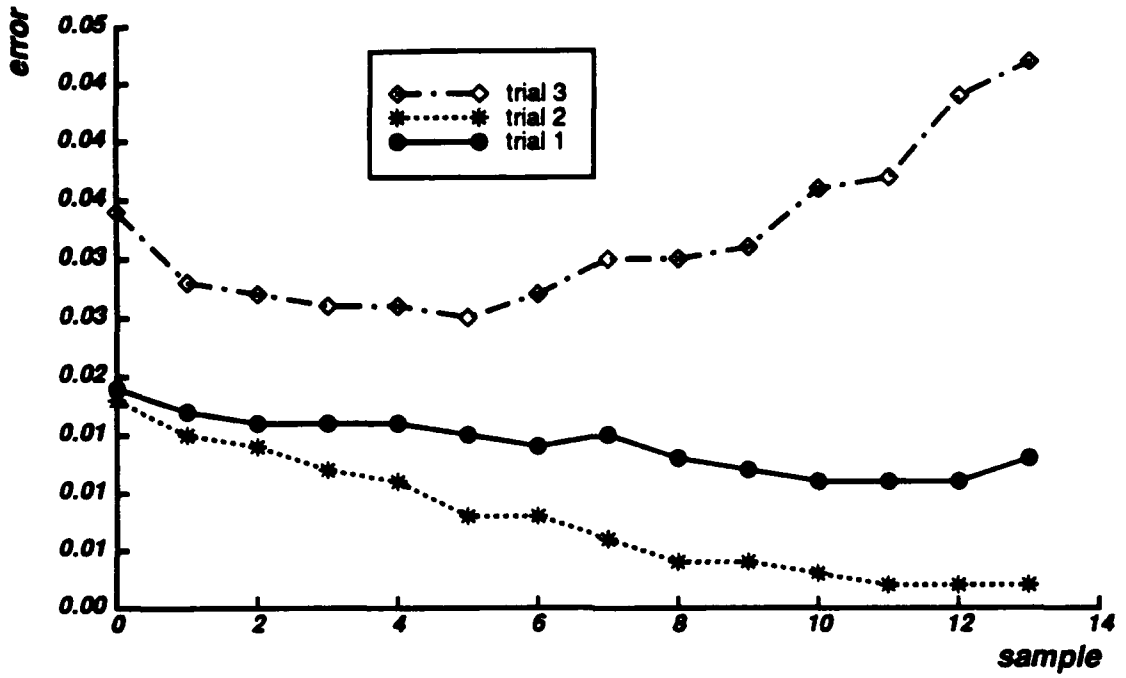


Figure 5.6: *Plot of error term, by sample. This plot shows the error terms, by sample, returned by the minimization for the three trials shown above.*

2.2N.

5.5.3 Improving Contact Normals

The accuracy of the results obtained can be further verified by examining the corrected surface normals. The grasp used in the experiments had two fingers (tips 1 and 2) opposing the thumb (tip 3). The corrected surface normals should reflect this, and oppose each other. Table 5.3 shows the average surface normal measurements obtained before and after tip orientation correction. There is again a noticeable improvement in the results. If the angle between vectors is used as a measure, then tips 1 and 2, which should be parallel, show a difference of 0.23 radians before correction and 0.19 radians after correction. Tips 1 and 3, which should oppose each other, show a difference of 2.61 radians before correction and 2.93 radians after, and tips 2 and 3, which also oppose

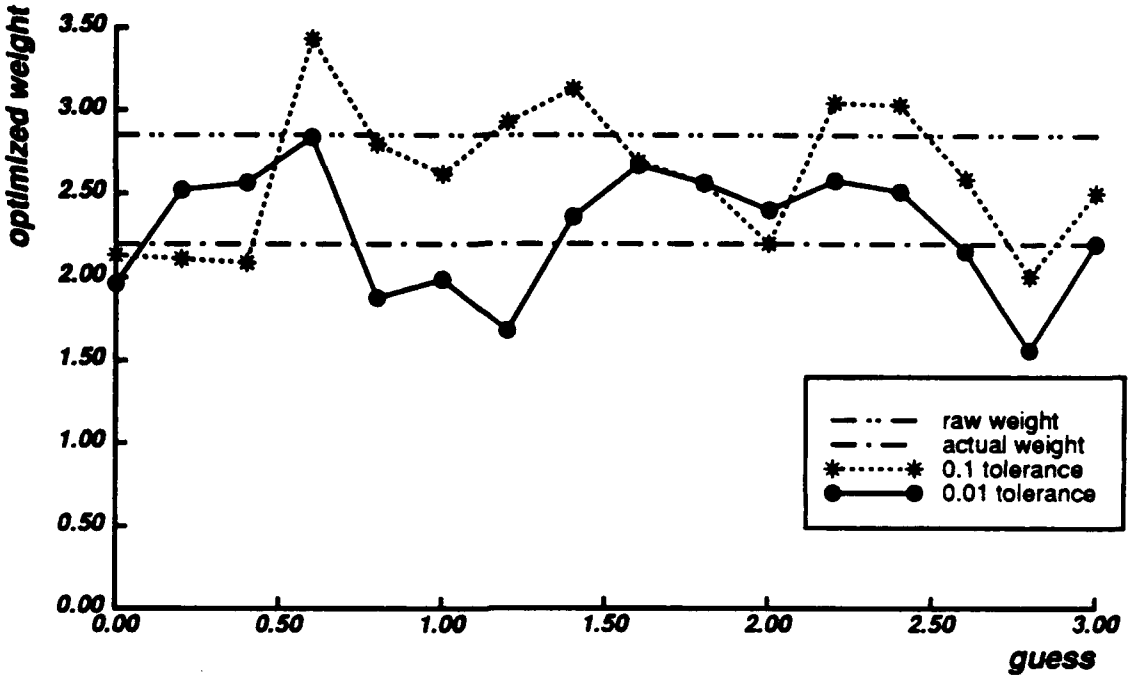


Figure 5.7: *Variation based on initial guess. This plot demonstrates the variation in final weight convergence value, based on the initial weight guess.*

θ	Tip 1	Tip 2	Tip 3
x	0.54	0.70	-0.21
y	0.54	0.51	-0.29
z	0.64	0.50	-0.93

θ	Tip 1	Tip 2	Tip 3
x	0.64	0.73	-0.47
y	0.28	0.34	-0.36
z	0.71	0.59	-0.81

Table 5.3: *Tip normal directions before and after corrections. The average normal directions for the first of the trials is shown both before (left) and after (right) the rotation corrections have been applied.*

each other, show a difference of 2.43 radians before correction and 2.80 radians after.

5.6 Simulations

Simulations were conducted to gain some insight into this somewhat disappointing experimental performance. A model that predicts the forces felt by the fingertip sensors is used to generate synthetic data. By using this model, a more rigorous investigation of the refinement process is possible. Various controlled simulations will examine the convergence properties of the numerical methods under differing conditions. The accuracy of the small angle linearization method will also be examined.

An important question that has not yet been answered is whether pose refinement is possible at all. Both the numerical minimization method and the small angle linearization method make the assumption that by reorienting a grasped object an independent set of sensor readings can be obtained. Some lurking dependencies in the measurement might make it impossible to solve for the orientation corrections. Through intuition, it has already been mentioned that a world rotation around the z axis cannot be recovered. Perhaps there are other parameters of the correction that also cannot be found? An argument can be made that shows by reorienting the hand it is possible to find the point where each fingertip is aligned with the gravity vector. This gives hope that fingertip corrections can be found by using more general motions of the hand. The simulations in this section will show that, aside from the world z rotation, all fingertip orientation parameters are deducible from multiple readings of the sensor data.

5.6.1 Simulator Design

Before discussing the simulation experiments, a model for predicting fingertip grasping forces is presented. This model is used to obtain the synthetic data used in the simulations. Refer to Figure 5.8 for the notation used in the subsequent derivation.

The grasped object is assumed to be in static equilibrium, with gravity as the only external force. Without loss of generality, the coordinate system is assumed to be at the

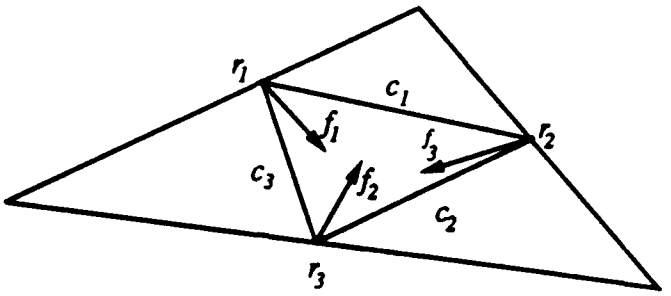


Figure 5.8: *Grasping force model. The grasping forces, f_i, can be found using torque and force balance equations, along with a set of internal forces c_i.*

object's center of mass. The force and torque balance equations are written as

$$m\vec{g} = \sum_{i=1}^{3} f_i \tag{5.18}$$

$$0 = \sum_{i=1}^{3} r_i \times f_i, \tag{5.19}$$

where mg is the object's weight, r_i is the vector to the fingertip contact point, and f_i is the fingertip contact force. For simulation purposes, the object weight and contact points are specified. The contact forces are desired. Since Equations 5.18 and 5.19 can each be written as x, y, and z component equations, there are six equations and nine unknowns. The three additional parameters that must be specified are commonly referred to as the internal forces, and can be defined as follows:

$$\begin{aligned} c_1 &= r_{12} \cdot (f_1 - f_2) \\ c_2 &= r_{23} \cdot (f_2 - f_3) \\ c_3 &= r_{31} \cdot (f_3 - f_1). \end{aligned} \tag{5.20}$$

Note that these internal force equations are arbitrary. Any three independent equations that relate the fingertip forces can be used. These particular equations capture the notion that the internal forces are the amount of squeezing force between each of the fingers.

Equations 5.18, 5.19, and 5.20 can be written in matrix form as,

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & -r_{1z} & r_{1y} & 0 & -r_{2z} & r_{2y} & 0 & -r_{3z} & r_{3y} \\ -r_{1z} & 0 & r_{1x} & -r_{2z} & 0 & r_{2x} & -r_{3z} & 0 & r_{3x} \\ -r_{1y} & r_{1x} & 0 & -r_{2y} & r_{2x} & 0 & -r_{3y} & r_{3x} & 0 \\ r_{12x} & r_{12y} & r_{12z} & -r_{12x} & -r_{12y} & -r_{12z} & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{23x} & r_{23y} & r_{23z} & -r_{23x} & -r_{23y} & -r_{23z} \\ -r_{31x} & -r_{31y} & -r_{31z} & 0 & 0 & 0 & r_{31x} & r_{31y} & r_{31z} \end{bmatrix} \begin{bmatrix} f_{1x} \\ f_{1y} \\ f_{1z} \\ f_{2x} \\ f_{2y} \\ f_{2z} \\ f_{3x} \\ f_{3y} \\ f_{3z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \\ 0 \\ 0 \\ 0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} . \quad (5.21)$$

This set of linear equations can be easily solved to find the fingertip forces.

To obtain a set of simulated sensor readings as an object is rotated, the object weight mg, the internal forces c_i, and the contact vectors r_i, are specified. To rotate the object, the r_i vectors are subjected to a rigid rotation. The new r_i vectors are then used to compute the next set of fingertip force readings. This process is repeated to obtain the desired number of force samples. This simulation gives forces that would correspond to those generated by motion of a robotic arm with the hand's finger positions fixed.

This scheme assumes that the internal forces c_i are constant throughout the rotation. In reality, these forces are a function of the control law used to servo the fingers. It can

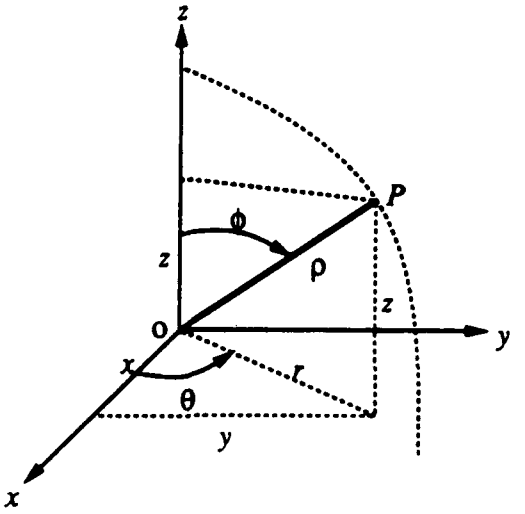


Figure 5.9: *Spherical coordinate system. By varying ϕ and θ, a rotational error of magnitude ρ can be generated in a particular direction.*

be shown that the constant force assumption holds if a proportional controller is used to drive each fingertip to a set point somewhere inside the object. This corresponds to how the hand used in the experiments was controlled.

In the final step for obtaining simulated data for the refinement process, the synthesized force readings for each fingertip are rotated by a fixed amount. This rotation is the error that the refinement process must recover. For some of the simulations, random noise and bias offsets are also added to the readings. This helps investigate the robustness of the numerical method.

5.6.2 Orientation Error Direction

To determine if the orientation of the fingertip error had an effect on the performance of the correction method, the following experiment was performed. A sampling of all possible error orientations was applied to a set of synthetic data. For each error orientation, a correction was computed using the minimization algorithm. The error between

the actual correction and the computed correction was then computed. Error is defined to be the magnitude of the difference in θ values of the equivalent axis rotations for the actual and computed orientations. The space of possible error directions is computed using a two parameter spherical coordinate system, as shown in Figure 5.9. If the magnitude of the error is ρ, the parameters ϕ and θ can be sampled to obtain a set of error vectors,

$$\begin{aligned} x &= \rho \sin\phi \cos\theta \\ y &= \rho \sin\phi \sin\theta \\ z &= \rho \cos\phi \end{aligned} \tag{5.22}$$

where $[xyz]^T$ are the α, β, γ error angles.

Using the spherical system defined in Figure 5.9 and Equation 5.22, the convergence characteristics of a fixed single tip error directed in all possible orientations was examined. The plot in Figure 5.10 shows the results of this simulation. The x and y axes represent values of ϕ and θ from 0 to 2π. The z axis represents the magnitude of the ratio between the actual and recovered orientation corrections. Note that in almost all directions, the method significantly improves the normal direction measurement. While in this example certain directions did not show as impressive an improvement as other directions, this was more a function of the particular samples, rather then a systematic failure of the method in a particular direction.

The previous results were obtained from perfect data. In reality, at least two types of problems with the sensor data can occur. The readings are subject to random noise and to systematic calibration errors. A number of simulation experiments were performed to investigate how well the method performs under these more realistic assumptions, and are described in the next several section.

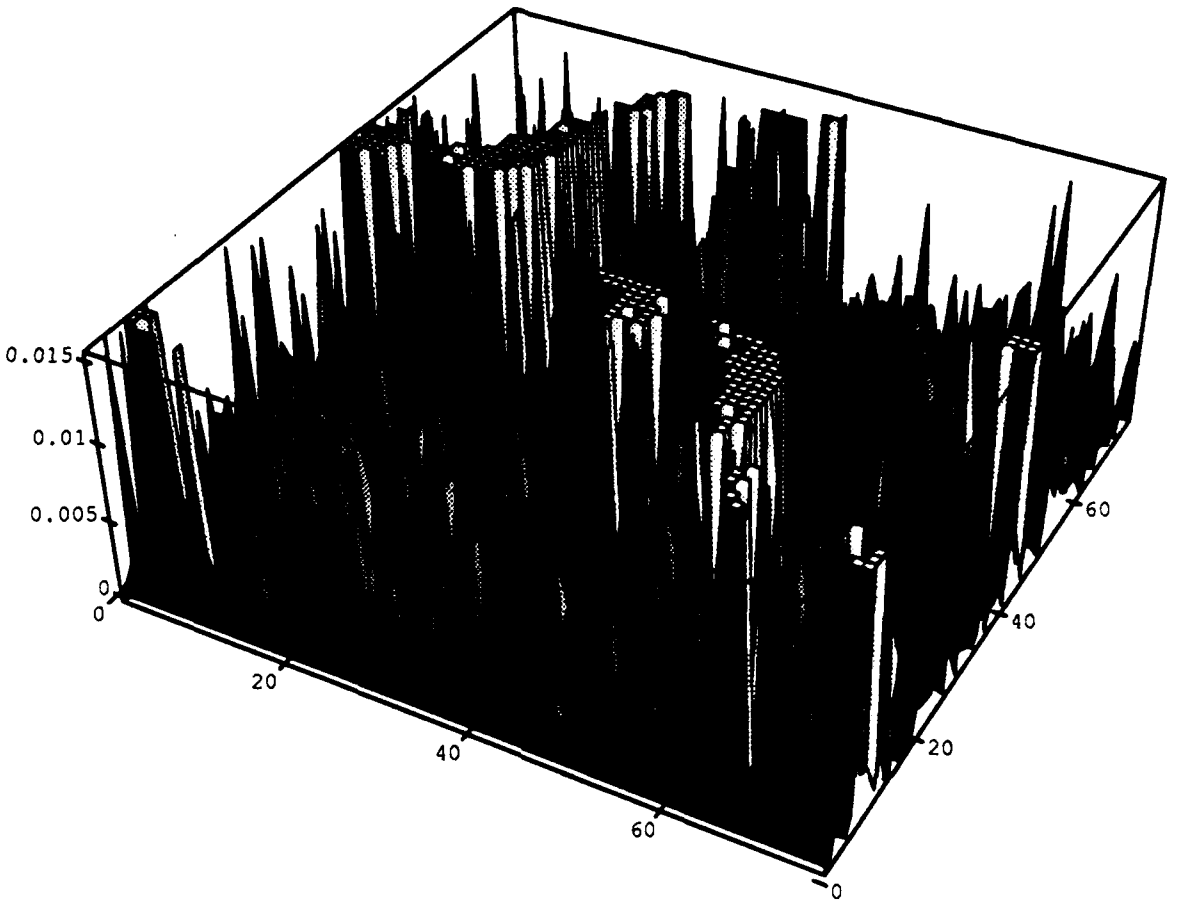


Figure 5.10: *Convergence error versus applied error direction. The magnitude of the plot indicates the relative performance of the algorithm for the given phi and θ. The lower the value, the better the algorithm performed.*

5.6.3 Sensor Noise

Table 5.4 presents the eight actual and recovered fingertip correction angles for simulated sensor data with varying noise levels. The angles, listed in order, are the zyx corrections for each finger. The first column shows the actual error applied to the eight recoverable fingertip orientation angles. Each subsequent column shows the correction recovered, given the indicated noise level. Note that the method is able to recover the correction accurately, even with levels of random noise exceeding 25 percent. To a certain extent this is to be expected, since on average the noise cancels itself out. Nonetheless, it is

actual error	percent noise 0	5	10	15	20	25	30	35	40	45	50
-0.1	-0.10	-0.10	-0.10	-0.11	-0.11	-0.11	-0.11	-0.12	-0.13	-0.14	-0.20
-0.2	-0.20	-0.18	-0.15	-0.12	-0.10	-0.09	-0.07	-0.03	-0.00	0.05	0.32
-0.1	-0.10	-0.10	-0.11	-0.13	-0.12	-0.14	-0.14	-0.13	-0.13	-0.13	-0.08
0.1	0.12	0.12	0.10	0.09	0.10	0.09	0.09	0.11	0.11	0.12	0.09
0.2	0.19	0.20	0.19	0.20	0.20	0.19	0.19	0.20	0.21	0.22	0.29
-0.2	-0.18	-0.18	-0.20	-0.22	-0.22	-0.24	-0.25	-0.23	-0.23	-0.21	-0.09
0.3	0.27	0.27	0.26	0.25	0.27	0.26	0.27	0.30	0.31	0.33	0.28
-0.2	-0.24	-0.21	-0.20	-0.16	-0.15	-0.14	-0.13	-0.09	-0.05	0.02	0.44

Table 5.4: *Effect of noise on recovered angles. This table shows the recovered eight correction angles, with varying amounts of noise added to the simulated data. The rows of the table show the correction angles for each of the recovered corrections.*

recovered weight	percent noise 0	5	10	15	20	25	30	35	40	45	50
optimized	100.0	100.0	99.8	99.9	99.3	98.4	97.9	98.0	98.0	99.4	114.2
average	100.0	99.8	99.1	98.8	98.0	97.0	96.4	96.0	95.7	96.2	107.3

Table 5.5: *Effect of noise on object weight. This table shows the recovered object weight, with varying amounts of noise added to the simulated data. The first row shows the optimized weight, the second shows the average weight.*

promising that the numerical minimization is able to converge to the correct solution even when substantial sensor noise is present.

Table 5.5 presented the actual and recovered object weight for simulated sensor data with varying noise levels. For these trials, the object weight was assumed to be unknown, and was included in the minimization error term. An initial guess of zero was used. In the table, the term *optimized* weight refers to the actual weight recovered by the minimization error term. The *average* weight refers to an average over all samples of the corrected weights. The optimized weight in general provides a more accurate solution. Notice that for even a large 45 percent noise, the weight is reliably recovered.

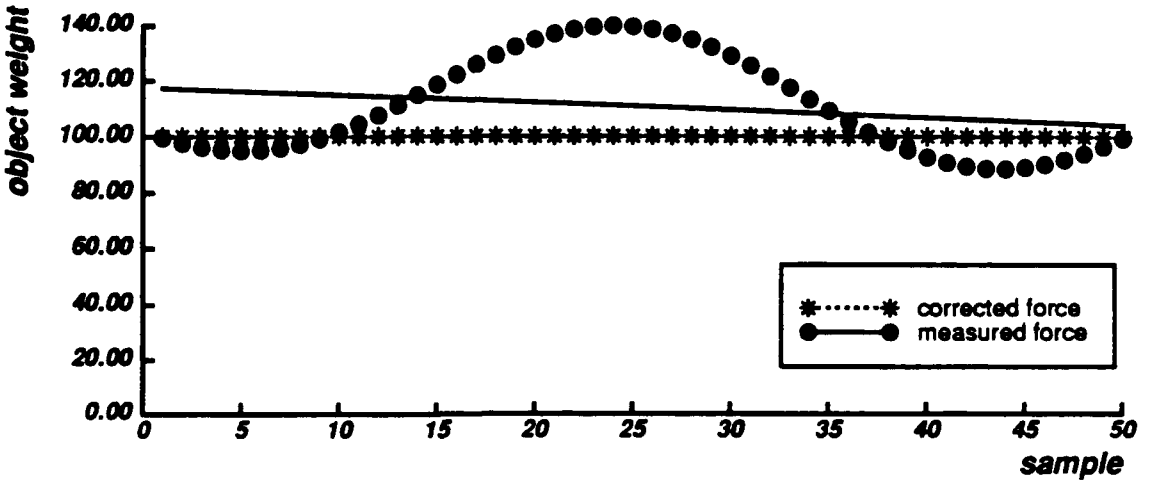


Figure 5.11: *Recovered object weight by sample, without noise. The raw and corrected object weights are plotted, by sample, in this diagram. The best fit line for each set of samples is also shown.*

Figure 5.11 and Figure 5.12 show plots of object weight (sum of the z forces) for each sample from a particular trial. For these experiments, 50 samples were used. The object weight was 100N. The raw z force is plotted in dots, the corrected z force is plotted in stars. For each plot, the best fit line to the points is also shown. The plot shown in Figure 5.11 shows the results for simulated data without noise. The plots shown in Figure 5.12 shows the results for simulated data with 10, 20, and 30 percent added random noise. Notice that in all cases the method can accurately recover the correct weight.

5.6.4 Orientation Error Magnitude

Simulations were performed to investigate how the magnitude of the fingertip orientation error effects the performance of the correction method. For error magnitudes ranging from 0.1 to 1.4 radians, 625 trials were performed, where the error was varied across all possible directions. In addition, random noise ranging from 10 to 50 percent of the signal was added. Figure 5.13 plots the results of this experiment. In the plots, the

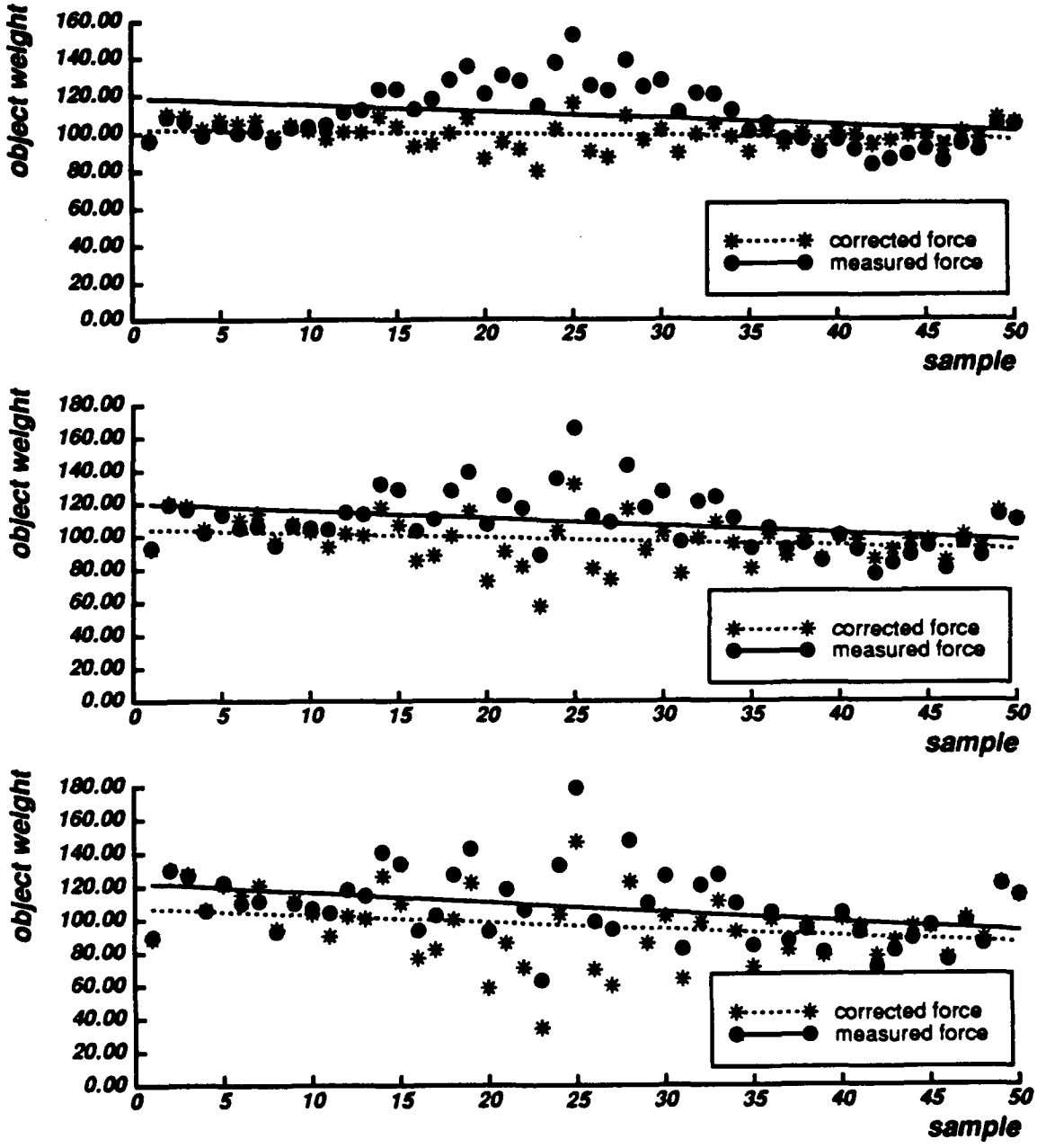


Figure 5.12: *Recovered object weight by sample, with noise. The raw and corrected object weights are plotted, by sample, in this diagram. The best fit line for each set of samples is also shown. From top to bottom, noise of 10, 20, and 30 percent has been introduced into the data.*

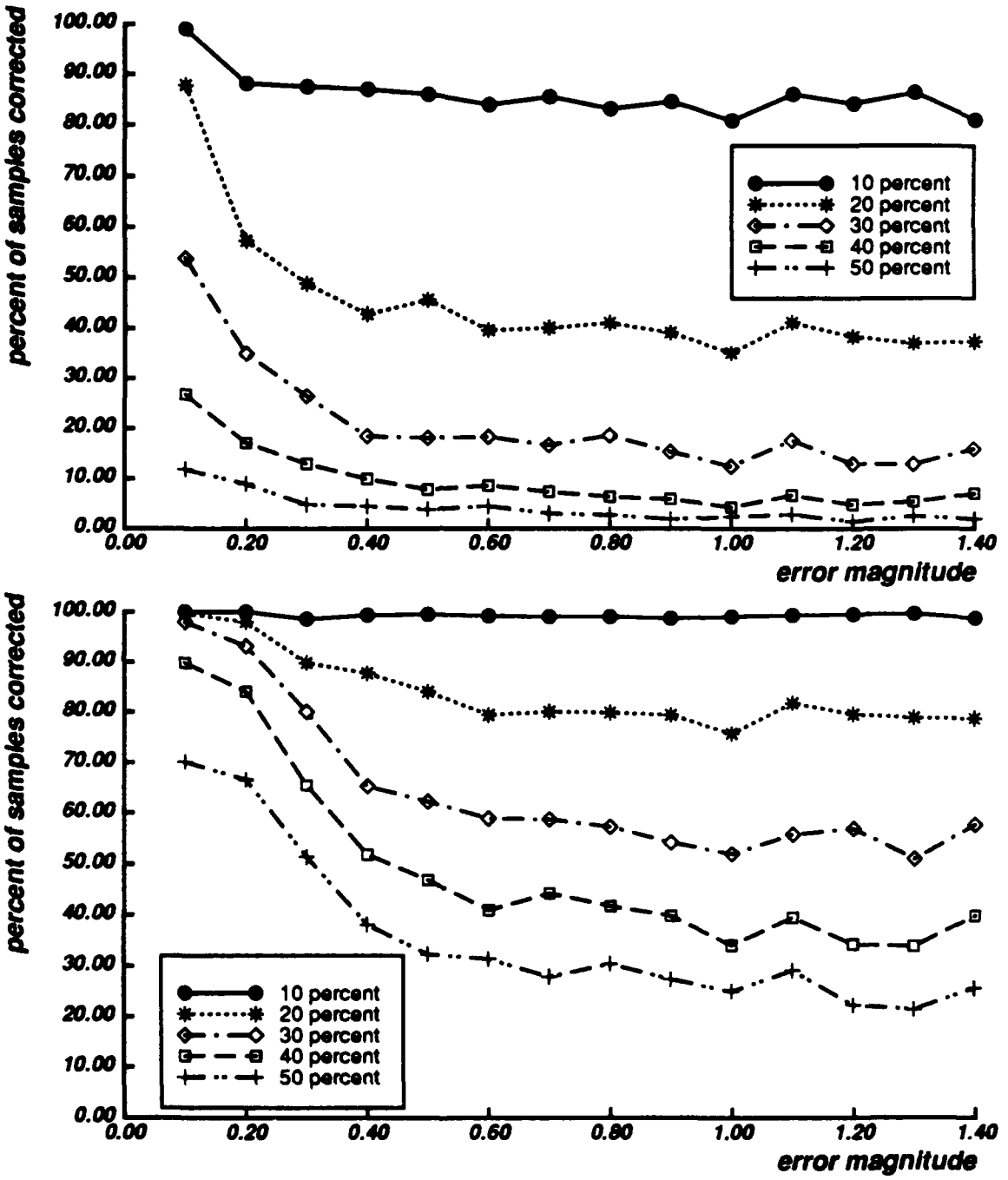


Figure 5.13: *Percent convergence with noise. A constant error magnitude is sampled in all directions. For these samples, the percent that show a significant correction is computed. The percentage is plotted against the error magnitude. The upper plot uses a correctness threshold of 0.1 radians, the lower plot uses 0.2 radians.*

percent of samples corrected are the percent of trials that are corrected to with 0.1 and 0.2 radians of the actual normal direction.

5.6.5 Sensor Calibration Bias

Simulations were performed to investigate how sensor biases affect the performance of the correction method. Biases of 5, 10, and 15 percent was added to the sensor readings. In addition, random noise of 10 and 20 percent was added. Figure 5.14 plots the results of the experiment with 10 percent noise, while Figure 5.15 plots the results with 20 percent noise. Notice that even small biases dramatically reduce the performance of the method. As will be discussed later, this problem was partly responsible for degrading the experimental performance that was observed.

5.6.6 Small Angle Approximation

Figure 5.16 examines the sensitivity of the linearized refinement algorithm to the small-angle approximation that it uses. The figure plots the magnitude of the correction error against the percent deviation of the correction. Percent deviation is defined to be the ratio of the correction error to the applied error. Each data point is computed from a number of trials each at the given error magnitude, in different directions. An acceptable correction deviation is obtained for errors under 1×10^{-2} radians. Above that, the small angle approximation used to linearize the problem fails, and the corrections obtained are no longer accurate.

5.7 Summary and Discussion

This chapter presented a method for continuously calibrating the orientations of the fingertips of a hand to obtain more accurate measurements of contact normals and forces. The methods are needed because calibration errors in the robot's kinematics significantly

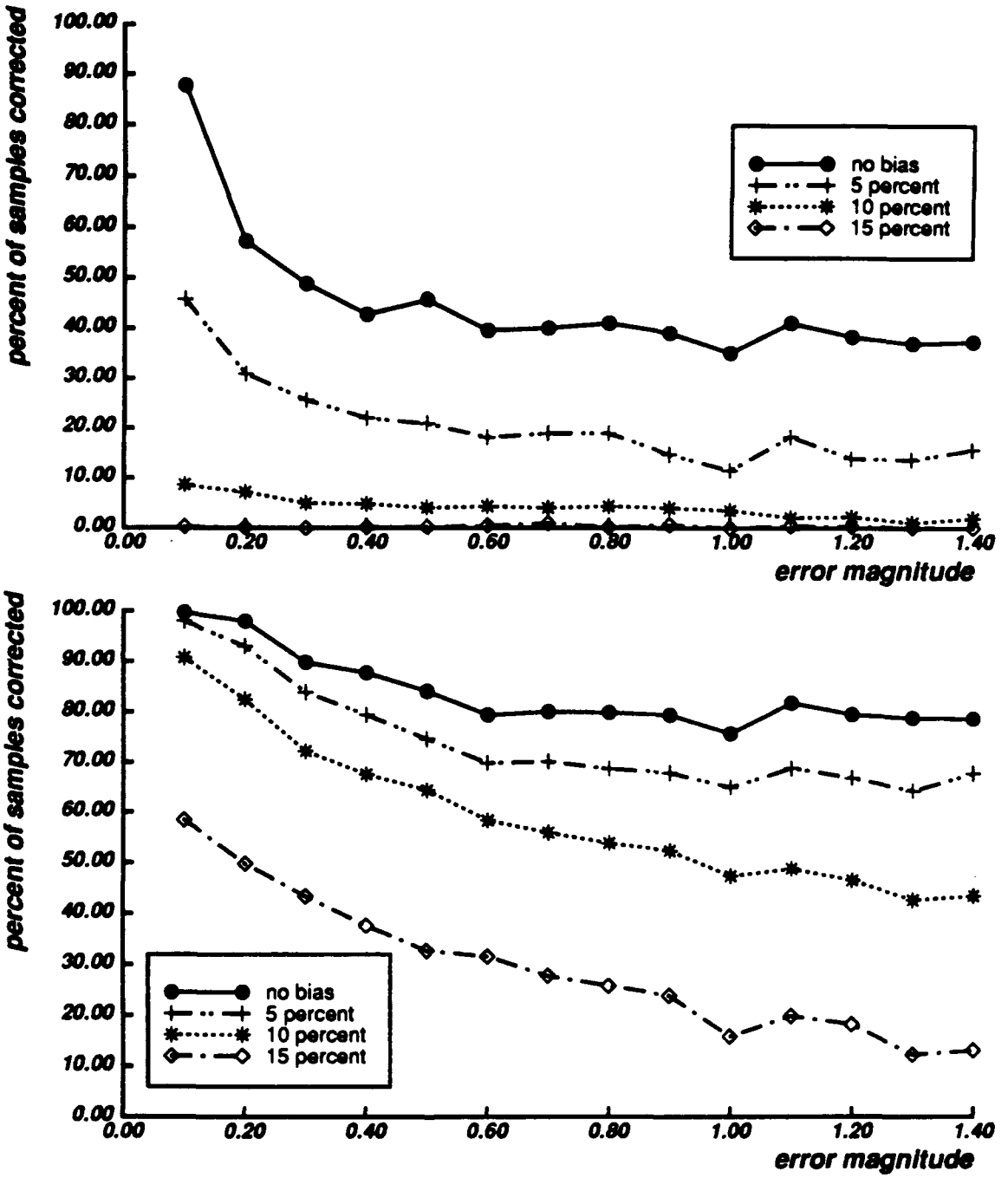


Figure 5.14: *Percent convergence with 20 percent noise and bias. A constant error magnitude is sampled in all directions, with a fixed sensor bias added to the simulated readings. For these samples, the percent that show a significant correction is computed. The percentage is plotted against the error magnitude. The upper plot uses a correctness threshold of 0.1 radians, the lower plot uses 0.2 radians.*

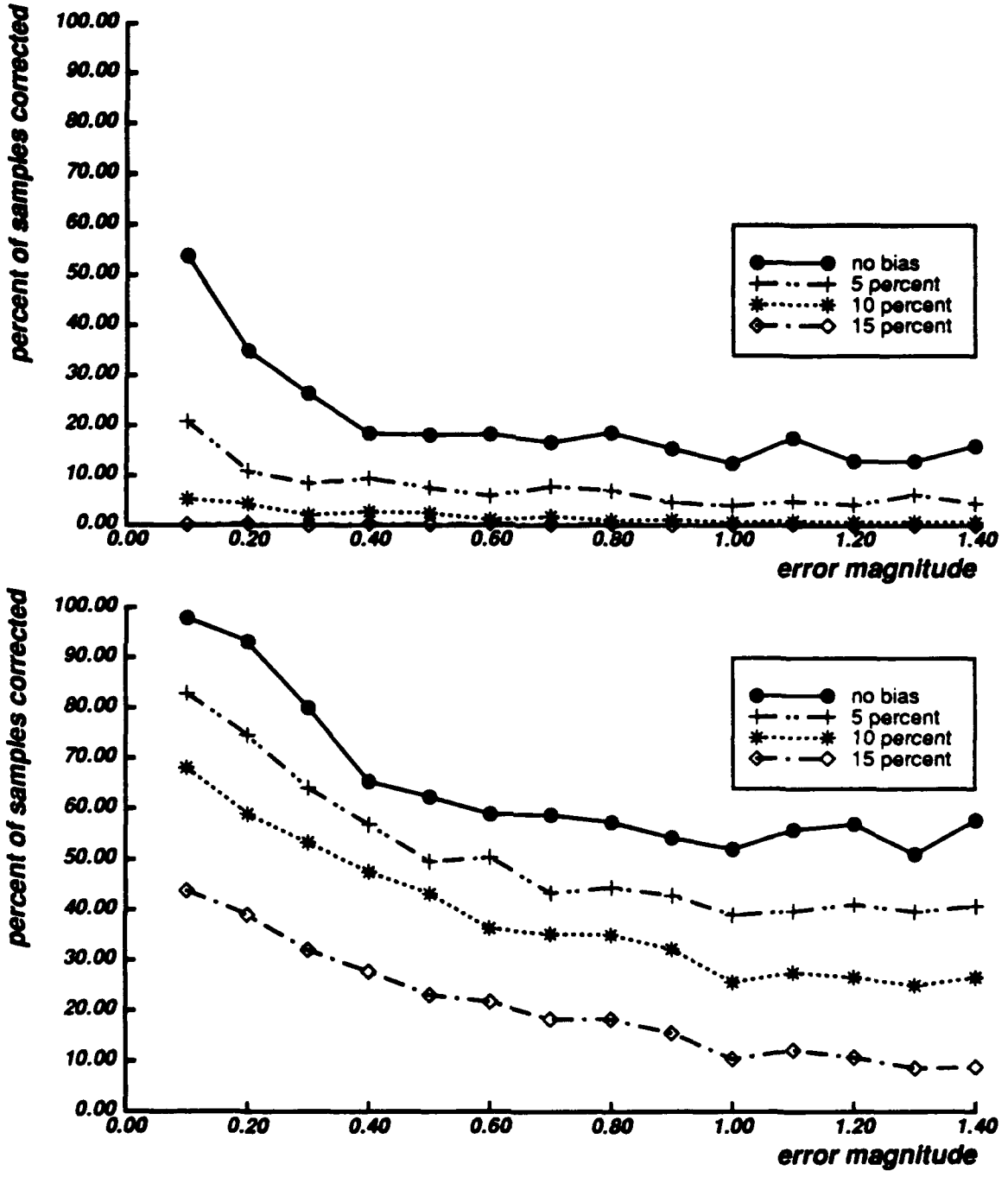


Figure 5.15: *Percent convergence with 30 percent noise and bias. A constant error magnitude is sampled in all directions, with a fixed sensor bias added to the simulated readings. For these samples, the percent that show a significant correction is computed. The percentage is plotted against the error magnitude. The upper plot uses a correctness threshold of 0.1 radians, the lower plot uses 0.2 radians.*

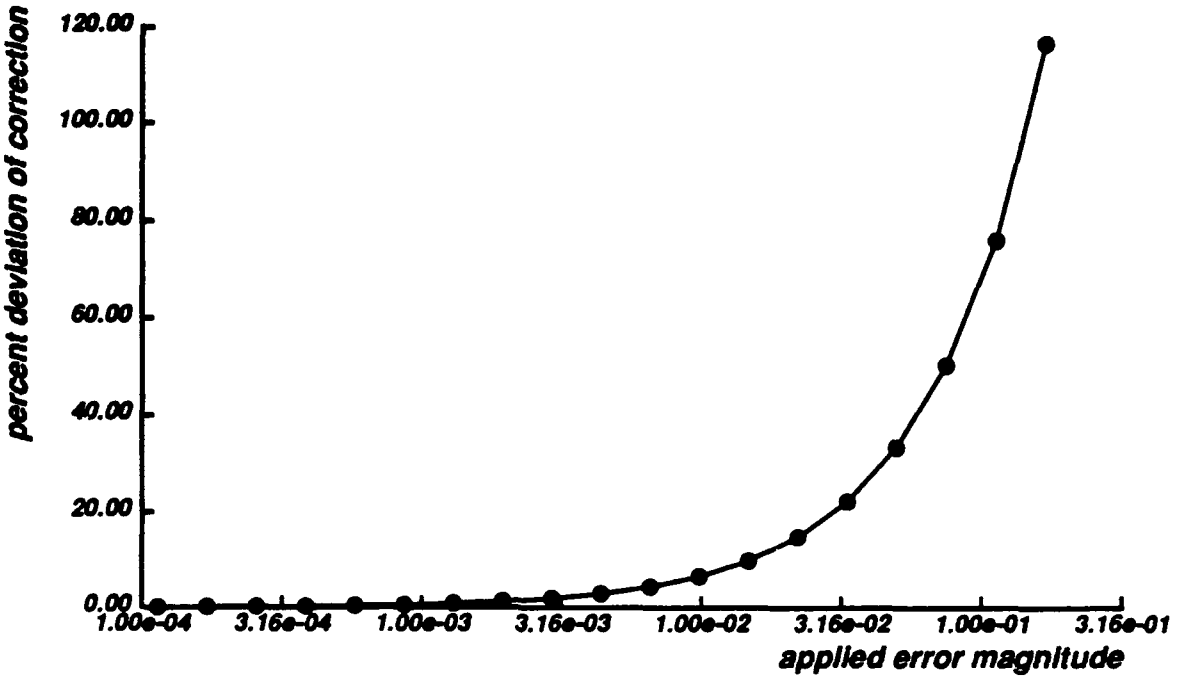


Figure 5.16: *Percent deviation of linearized correction. The x-axis denotes the applied error magnitude, plotted in a logarithmic scale. The y-axis shows the percent deviation of the correction for the given error.*

reduce the utility of the fingertip sensors. Without good global knowledge of forces and normals, certain recognition and manipulation tasks are very hard to perform.

The calibration method discussed uses multiple sensor readings obtained while the robotic arm is moving a hand holding a grasped object. The fingertip to world coordinate frame transform is refined according to the constraint that the sum of the fingertip forces must always equal the object's weight.

The simulations and experiments conducted gave mixed results. In general, they indicated that the method works well for certain types of errors. For large random noise the method was able to recover the correct fingertip orientations. On the other hand, for relatively small sensor bias errors, the performance deteriorated rapidly.

The experimental results were not particularly good. While some of the trials produced good results, with an accuracy close to 5 percent, others trials had much worse

performance. Using the results of the simulations, three potential causes for the relatively poor experimental performance were identified. The next sections discuss them, and propose potential solutions.

5.7.1 Poor Sensor Calibration

The simulations indicated that while the method is relatively immune to noise, it is susceptible to calibration and other bias problems. Upon further investigation, the actual calibration of the fingertip sensors proved to be rather poor. The calibration process requires mounting the sensor on a stand, and applying known forces at particular points on the device. Unfortunately, the apparatus for applying these forces was rather crude. At first it was hoped that a calibration accuracy of 5 percent would be possible. In actually, the accuracy was probably no better than 15 percent.

5.7.2 Inadequate Sample Size

The simulations confirmed that the method's susceptiblity to noise is greatly reduced as the number of samples is increased. For the noise levels introduced in the simulations, 50 samples proved to be necessary. Due to limitations in the experimental procedure, only 15 samples were used, apparently too few to mitigate the effects of the noise levels that were present. This alone would greatly degrade the convergence characteristics even at moderate noise levels.

5.7.3 Incorrect Assumptions

Finally, poor experimental performance could be a result of incorrect assumptions. The most critical assumption that was made is that all the error in sensor orientation was a result of calibration errors in the hand. Motion of the arm was assumed to be accurate. While all indications are that this assumption was valid, further investigation on this point is warranted.

Thus, while from a theoretical standpoint the refinement method presented showed promise, various experimental limitation reduced its performance. Future work should be conducted to better understand these problems, and to help identity improvements, as previously suggested.

Discussion

Chapter 6

This report examined the problem of finding the pose of an object grasped by a hand. The methods that were studied can successfully determine an object's pose using a minimum of information. The use of kinesthetic sensing, along with knowledge of the grasp acquisition strategy employed by the hand, usually provided sufficient data for the task. In the case of the pose determination problems conducted with the Utah-MIT hand, just 16 numbers were used as input to the recognizer. The power of these methods can be attributed to their careful exploitation of the constraints inherent in the problem.

Pose determination was argued to be an important problem for several reasons. For almost all manipulations, at least a certain amount of information on the location of objects both relative to the robot and relative to the world is required. Performing an accurate calibration is not enough, as error and uncertainty is unavoidable. Even strategies that are guaranteed to work usually have a bound on the error in object position tha: they can tolerate. The reality of robotics is that uncertainty is unavoidable. Pose determination provides a way to compensate for this uncertainly.

Some typical problems that are suitable for the pose determination methods studied

in this report include parts alignment, grasp verification, and haptic exploration:

- *Parts alignment:* A simplified multi-linked gripper operating on an assembly line could grasp a part, verify its pose, and then direct another robot to the part in the now known location. Thus, the gripper serves both as a position sensor and a clamp, much the same role as a conventional parts feeder. An advantage with this approach is that the system is not particular to the part. Retooling for new parts is simplified.

- *Grasp verification:* When a dextrous hand has completed a grasp, position uncertainty in the system usually causes in a wide variation in the object's final pose. Pose determination could be used to find the position, and then used to plan subsequent manipulations around the true object pose, rather than the inaccurate planned pose.

- *Haptic exploration:* Hands, both human and robotic, can be thought of as sensory organs. It is possible for a hand to provide the sensory information necessary for a wide variety of recognition tasks. These exploratory motions are used for tasks such as reaching into a bag and pulling out a particular object and groping in the dark for a telephone receiver. The pose determination methods studied provide certain insight to the information sources that human haptics must utilize.

The methods presented in this report show that hand shape and knowledge of the grasp acquisition strategy can provide useful information for solving these types of problems.

6.1 Review of the Report

The first algorithm, presented in Chapter 3, studied a method to find object poses that were consistent with a hand shape. Assignments of object vertices to finger edge segments were examined in a systematic manner, using an interpretation tree to guide the search. By carefully pruning inconsistent vertex-edge pairs, the portion of the tree

that was examined was minimized. This method found all potential placements of the object that were consistent with the hand shape, given certain contact assumptions. Generated poses were then verified according to several criterion. Most importantly, the pose and hand shape were checked to insure compatibility with the grasp acquisition strategy that was used.

Next, Chapter 4 showed how a memory could be used to speed pose determination. By storing past experience in the memory, the on-line determination process can be reduced to a simple lookup operation. To improve accuracy, regression analysis was used on similar poses, providing a way to interpolation between memory entries. The use of regression also allowed the memory to be compacted. The interpolation methods were general, and did not depend on solutions that are particular to specific objects or hands.

Finally, Chapter 5 examined how additional information provided by contact sensors could be used to refine a pose estimate. The pose of an object model can be fit to measured fingertip surface normals. The fitting process is only as accurate as the surface normals are themselves. Since sensors measure contact in a local coordinate frame, and since the fitting requires global data, accurate kinematic models and kinesthetic data is also required. In practice, calibration errors were found to greatly degrade the potential performance of this type of method. The chapter explored a refinement process that attempted to correct for calibration errors using world invariants.

6.2 Hand Design for Haptics

It is interesting to explore how the findings of this report could be of use to hand designers. This section briefly addresses this problem, in particular by exploring the notion that a hand should be designed as much for recognition as for manipulation.

The role of hand shape for recognition, as used in this report , is twofold. It is used for both pose generation and for pose verification.

Pose generation relies on the existence of a minimum number of contacts between the object and the hand. By searching the space of possible contact assignments, potential object poses are found. Because a certain minimum number of contacts are necessary, the hand must have enough fingers and links to usually achieve the necessary number of contacts in a grasp.

Pose verification relies heavily on the shape of the hand. The enclosure formed by the hand around the grasped object is tested for intersection against the postulated object position. A hand that forms a better enclosure will usually permit fewer collision free object placements.

Both the generator and verifier stages of the algorithm benefit from hands with more fingers and links. However, additional links increase the contact assignment search space. The simulations in Section 3.7 suggest that for three-dimensional problems the combinatorics are rather large without some contact information. Those results indicate that the search time is greatly reduced simply by sensing if contact has been made with a particular link, without knowledge of the contact location. Further search reductions can be obtained either by reducing the link lengths, or by using more precise contact sensing.

Experiments performed in Section 4.5.2 suggest that the the mapping from hand shape to object pose benefits from the addition of simple contact sensor information. Without any contact sensing, hand shapes frequently map to more than one object pose. For the cases where a unique mapping does not exist, unambiguous pose determination cannot be performed. If the hand links that are in contact with an object are known, the results from that section suggest that the mapping ambiguity is greatly reduced. The additional sensing power obtained by adding this type of sensing is larger than that obtained by adding a small number of additional links to each finger. This suggests to designers that adding minimal contact sensing, rather than just more finger segments, may be the best way to improve the recognition power of a hand.

While it is clear that additional sensor information will always be helpful for pose determination, these results indicate that very little additional sensor information can be put to good use. In particular, the type of sensing information that has been shown to be useful is easy to obtain. Devices that can determine if contact with a link has been made can be reliably fabricated using existing sensor technology (see Section 2.3).

Using the methods developed in this report , pose determination without these basic sensors suffers from both a search combinatoric explosion, and from ambiguity in the results. Adding *minimal* amounts of additional sensors is likely to correct this.

6.3 Future Work

The work presented in this report provides a promising approach for solving the pose determination problem. Nonetheless, much work remains. A few of the more interesting problems that deserve attention are listed here:

- *Three dimensions:* The pose determination method presented in Chapter 3 was implemented in two dimensions. By examining the first stages of a full three-dimensional implementation, it was argued that this extension could be performed. The extension should be completed, and experiments conducted on data from dextrous hands.

- *Sensitivity:* A better understanding of the sensitivity to sensor noise is warranted. In particular, for the methods in both Chapters 3 and 4, bounds on the performance based on joint angle sensor performance should be developed. If the sensors are too noisy, potential poses would certainly be missed.

- *Sensor fusion:* A better investigation of how tactile contact sensors can be used with these methods would be worthwhile. For example, binary contact information, which would be easy to obtain, could be used to guide the tree search used in Chapter 3. The additional sensor information could be used to make the methods

more robust, and less sensitive to noise.

With additional work, the approaches studied in this report will not only give an understanding of the information content in a hand's shape and its grasp acquisition strategy, but they will lead to practical and robust methods for solving for the pose determination problem.

Experimental Apparatus

Appendix A

This appendix provides a detailed description of the hardware and software used by the experiments described in this report . Special attention is given to the components which the author helped design and implement. Section A.1 describes the setup for the constraint-based localization experiments from Chapter 3, Section A.2 describes the setup used for the memory-based recognition experiments from Chapter 4, and Section A.3 describes the setup for the sensor-based refinement experiments from Chapter 5.

A.1 The Utah-MIT Hand

A.1.1 Mechanical Design

The Utah-MIT hand [57] (Figure 2.2) was used for the constraint-based pose localization experiments that were described in Chapter 3. This hand has four fingers, each with four degrees of freedom. An anthropomorphic design was used, giving it much the same size and shape as a human hand. Each joint is connected to two tendons, one for extension and one for flexsion, giving the hand a total of 32 actuators. Specially designed

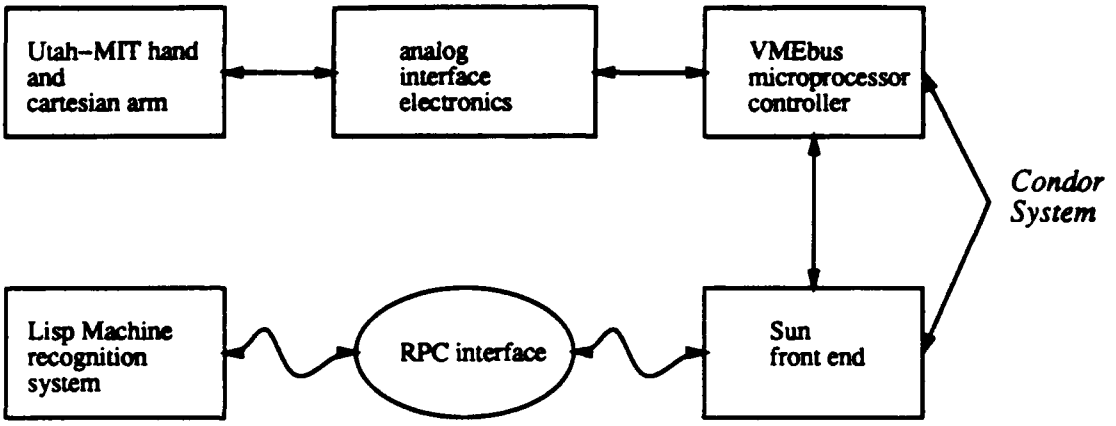


Figure A.1: *Block diagram of the Utah-MIT hand system, as used for the constraint based recognition experiments.*

pneumatic actuators are used for power, and are housed in an external actuator pack. The 32 tendons are routed from the actuator pack to the hand using a remotizer. This arrangement permits off loading the weight of the actuators from the hand itself, making mounting on existing robots easier.

The hand is mounted on a cartesian robot that provides three degree of freedom xyz motions. The hand itself is mounted on a gantry which provided xz motion. The remaining motions are provided by an xy positioning table. This setup gives a redundant motion in the x axis. The cartesian robot is actuated using stepper motors.

A.1.2 Control Architecture

The Condor [75] system is used to control the hand and cartesian robot. Condor is a real-time software environment designed for multiprocessor-based robotic control. The system provides interprocessor communication primitives, an efficient scheduler, and host computer support. A Sun workstation front-end, linked to the VMEbus multiprocessor backplane using a memory mapped connection, provides development support. The development tools include a symbolic debugger, a virtual terminal system based on

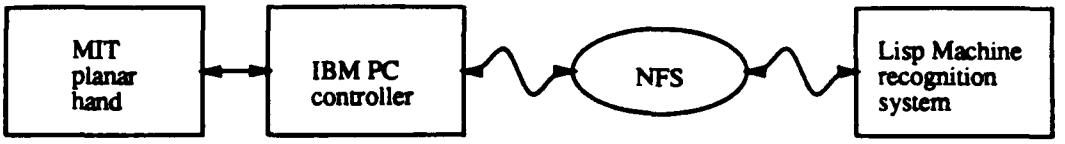


Figure A.2: *Block diagram of the MIT planar hand system, as used for the memory based recognition experiments.*

the X Window System, and a fileserver. Six Motorola 68020 processors are used in the system. The first processor is the system supervisor. The next four run the low level servo code, one processor for each finger. An additional processor is used to control the cartesian robot. The system achieves finger joint servo rates of up to 300 hertz.

A remote procedure call (RPC) interface to the hand-arm control system is also provided. The localization system is coded in Lisp and runs on a Symbolic Lisp Machine. The Lisp Machine communicates with the Condor using the RPC interface. Some of the operations that are supported by the RPC interface include commands to control the servo system, to enqueue trajectories, and to query joint positions and torques. Figure A.1 diagrams the system.

A.2 The MIT Planar Hand

A.2.1 Mechanical Design

This hand has two fingers, each with two joints. A belt pulley system connects the servo motors to the joints. Position sensing is provided by shaft encoders mounted on each motor. The finger surfaces are flat, providing a good surfaces for whole hand grasping. The hand is designed to allow reconfiguration of the separation distance between the fingers.

A.2.2 Control Architecture

The hand is interfaced to an IBM PC computer running PC/NFS, and uses a controller board based on the Hewlett-Packard HCTL-1000 servo processor. This chip provides low level servo control, along with the interface logic necessary to process the optical position encoder output from each joint. A control system was implemented that executed the grasping strategies used for building recognition memories. After executing a grasp, the resulting joint positions (the only sensory information required for recognition by the memory-based method) is written to a file on an NFS mounted disk. The recognition software is written in Lisp and runs on a Symbolics Lisp Machine. The Lisp Machine reads the joint data from the NFS mounted filesystem and finds consistent poses in real-time. Figure A.2 diagrams the system.

A.3 The Salisbury Hand

A.3.1 Mechanical Design

A Salisbury [90] hand, mounted to a Puma 500 robotic arm, was used for the refinement experiments described in Chapter 5. The hand has three fingers, each with three joints, as diagrammed in Figure 2.1. The actuator system uses $n+1$ motors for n joints. Thus, there are four motors for each finger, or twelve for the entire hand. Tendons connect the actuators to the joints.

The hand is equipped with Brock and Chiu [15] force sensing fingertips. Each of the fingertips has eight strain gauge sensors that are used to detected the contact forces and torques. The strain gauges are connected to analog amplifiers which are interfaced to the controller computer.

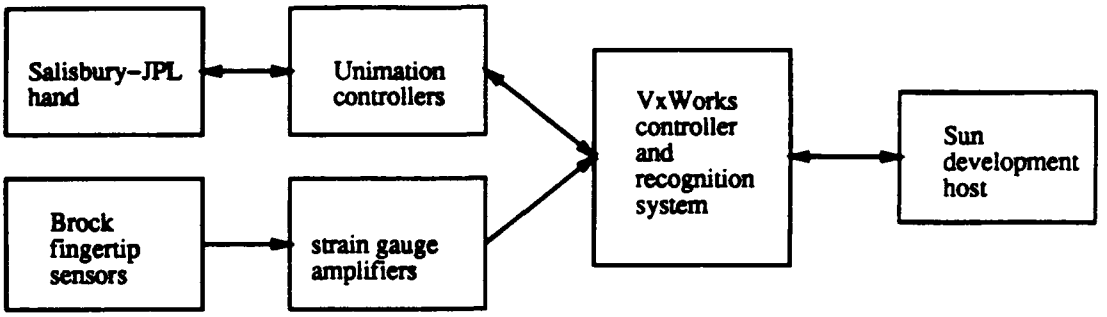


Figure A.3: *Block diagram of the Salisbury hand system, as used for the pose refinement experiments.*

A.3.2 Control Architecture

The control architecture for the Puma-Hand system is based on the VxWorks [108] real-time operating system running on Motorola 68030 processors. VxWorks provides a low-level kernel with a fast scheduler, networking tools, and a standard Unix-style library interface. The operating system provides NFS filesystem access to a host Sun workstation that is used for program development. A VMEbus backplane is used to interconnect three Motorola 68030 processors that each run VxWorks. Figure A.1 diagrams the system.

The Salisbury-JPL hand is interfaced to two Unimation servo controllers, which provide low level position control. The servo controllers are interfaced, via a parallel port, to the VxWorks system. A software module on one 68030 processor feeds position commands to the Unimation controllers. A higher level message-based trajectory controller, for enqueuing a sequence of position setpoints, provides the external software interface to the system.

The refinement code is written in C and runs on the VxWorks real-time system. The code communicates with the Puma, the hand, and the fingertip sensors. The hand is first commanded to close on an object until contact is detected by the fingertip sensors. The Puma then sweeps out a motion while the sensors are sampled. The refinement

process is then run on the collected data. Figure A.3 diagrams the system.

Bibliography

[1] J. Abramowitz, J. Goodnow, and B. Paul. The Pennsylvania articulated mechanical hand. In *Proceedings of ASME Conference on Robotics*, Chicago, 1983.

[2] P. K. Allen. *Robotic Object Recognition Using Vision and Touch.* Kluwer Academic Publishers, Boston, MA, 1987.

[3] P. K. Allen and R. Bajcsy. Object recognition using vision and touch. In *Ninth International Joint Conference on Artificial Intelligence*, pages 1131–1137, Los Angeles, CA, 1985.

[4] C. G. Atkeson and D. J. Reinkensmeyer. Using associative content-addressable memories to control robots. In P. H. Winston, editor, *Artificial Intelligence at MIT: Expanding Frontiers*, volume 2, pages 103–127. MIT Press, 1990.

[5] J. Barber, R. A. Volz, R. Desai, R. Rubinfeld, B. Schipper, and J. Wolter. Automatic two-fingered grip selection. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 890–896, San Francisco, CA, April 1986.

[6] R. E. Barnhill. Representation and approximation of surfaces. In J. R. Rice, editor, *Mathematical Software III*, pages 69–120. Academic Press, New York, NY, 1977.

[7] C. M. Bastuscheck. Area touch sensor for dextrous manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 151–156, Scottsdale, AZ, May 1989.

[8] S. Begej. An optical tactile array sensor. In *Proceedings of SPIE Conference on Intelligent Robots and Computer Vision*, pages 271–280, Cambridge, 1984.

[9] D. J. Bennett and J. M. Hollerbach. Self-calibration of single-loop, closed kinematic chains formed by dual or redundant manipulators. In *Proceedings 27th IEEE Conference Decision and Control*, pages 627–629, Austin, TX, December 1988.

[10] D. J. Bennett and J. M. Hollerbach. Closed-loop kinematic calibration of the Utah-MIT hand. In *Intl. Symposium Experimental Robotics*, Montreal, June 1989.

[11] A. Bicchi. Intrinsic contact sensing for soft fingers. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 968–973, Cincinnati, OH, May 1990.

[12] T. O. Binford. Sensor systems for manipulators. In *Proceedings of Conference on Remotely Manned Systems*, pages 283–291. Cal Tech, 1972.

[13] R. A. Boie. Capacitive impedance readout tactile image sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 370–379, Atlanta, 1984.

[14] L. Bologni, S. Caselli, and C. Melchiorri. Design issues for the U. B. robotic hand. In *NATO Advanced Workshop - Robots with Redundancy: Design, Sensing, and Control*, Salo, Italy, 1988.

[15] D. Brock and S. Chiu. Environment perception of an articulated robot hand using contact sensors. In *Proceedings of ASME Conference on Robotics*, pages 89–96, 1985.

[16] R. W. Brockett. Robot hands with rheological surfaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 942–947, St. Louis, 1985.

[17] M. Caporali and M. Shahinpoor. Design and construction of a five-fingered robotic hand. *Robotics Age*, 6:14–20, 1984.

[18] C. Z. Chammas. Analysis and implementation of robust grasping behaviors. Technical Report 1237, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1990.

[19] S. S. Checinski and A. K. Agrawal. Magnetoelastic tactile sensor. In *Robotic Sensors*, chapter 2, pages 229–235. Springer-Verlag, 1986.

[20] P. Chiarelli and D. De Rossi. Determination of mechanical parameters related to the kinetics of swelling in an electrically activated contractile gel. *Progress in Colloid and Polymer Science*, 48, 1988.

[21] P. Chiarelli and D. De Rossi. Progress in the design of an artificial uretheral sphincter. In *Proceedings of the 3rd Vienna International Workshop on Functional Electrostimulation*, Vienna, Austria, September 1989.

[22] K. J. Chun and K. D. Wise. A capacitive silicon tactile imaging array. In *Proceedings of IEEE International Conference on Solid-State Sensors and Actuators*, pages 22–25, 1985. VLSI based device.

[23] R. Cole and C. K. Yap. Shape from probing. *Journal of Algorithms*, 8(1):19–38. 1987.

[24] T. M. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:50–55, 1968.

[25] J. J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, 1986.

[26] M. R. Cutkosky, J. M. Jourdain, and P. K. Wright. Skin materials for robotic fingers. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1649–1654, Raleigh, NC, March 1987.

[27] P. Dario, A. Bicchi, F. Vivaldi, and P. C. Pinotti. Tendon actuated exploratory finger with polymeric skin-like tactile sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 701–706, St. Louis, 1985.

[28] P. Dario, D. De Rossi, C. Domenici, and R. Francesconi. Ferroelectric polymer tactile sensors with anthropomorphic features. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 332–340, Atlanta, 1984.

[29] M. R. Driels. Pose estimation using tactile sensor data for assembly operations. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1255–1261, San Francisco, CA, April 1986.

[30] R. E. Ellis. A multiple-scale measure of static tactile texture. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1280–1285, San Francisco, CA, April 1986.

[31] R. E. Ellis. A tactile sensing strategy for model-based object recognition. COINS Technical Report 87-96, University of Massachusetts, Amherst, MA, 1987.

[32] L. J. Everett and C. Y. Lin. Kinematic calibration of manipulators with closed loop actuated joints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 792–797, Philadelphia, PA, April 1988.

[33] R. S. Fearing. Basic solid mechanics for tactile sensing. *The International Journal of Robotics Research*, 4(3):40–54, 1985.

[34] R. S. Fearing. Some experiments with tactile sensing during grasping. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1637–1643, Raleigh, NC, April 1987.

[35] R. S. Fearing. *Tactile Sensing, Perception and Shape Interpretation.* PhD thesis, Stanford University, Stanford, CA, 1987.

[36] R. S. Fearing and T. O. Binford. Using a cylindrical tactile sensor for determining curvature. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 765–771, Philadelphia, PA, April 1988.

[37] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 15:1691–1704, 1980.

[38] P. C. Gaston and T. Lozano-Pérez. Tactile recognition and localization using object models: The case of the polyhedra on a plane. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):257–265, 1984.

[39] A. R. Grahn and L. Astle. Robotic ultrasonic force sensor arrays. *Robotic Sensors*, 2:297–315, 1986.

[40] H. Greiner. Passive and active grasping with a prehensile robot end-effector. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1990.

[41] W. E. L. Grimson. The combinatorics of local constraints in model-based recognition and localization from sparse data. *Journal of the ACM*, 33(4):658–686, 1986.

[42] W. E. L. Grimson and T. Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *The International Journal of Robotics Research*, 3(3):3–35, 1984.

[43] R. L. Ten Grotenhuis and T. N. Moore. Development of a carbon fibre based tactile sensor. In *Proceedings of SME Sensors '85 Conference*, Detroit, 1985.

[44] S. Hackwood, G. Beni, L. A. Hornak, R. Wolfe, and T. J. Nelson. Torque-sensitive tactile array for robotics. *The International Journal of Robotics Research*, 2(2):46–50, 1983.

[45] J. W. Hill. Touch sensor and control. In *Proceedings of Conference on Remotely Manned Systems*. Cal Tech, 1972.

[46] D. Hillis. *The Connection Machine*. MIT Press, Cambridge, MA, 1985.

[47] W. D. Hillis. A high-resolution imaging touch sensor. *The International Journal of Robotics Research*, 1(2):33–44, 1982.

[48] G. E. Hinton. Connectionist learning procedures. Technical Report CMU-CS-87-115, Carnegie Mellon University, Pittsburgh, PA, 1987.

[49] S. Hirose and Y. Umetani. The development of soft gripper for the versatile robot hand. In *Proceedings 7th International Symposium on Industrial Robots*, pages 353–361, Tokyo, October 1977.

[50] J. M. Hollerbach. A review of kinematic calibration. In O. Khatib, J. J. Graig, and T. Lozano-Pérez, editors, *The Robotics Review 1*, pages 207–242. MIT Press, Cambridge, MA, 1989.

[51] R. D. Howe and M. R. Cutkosky. Sensing skin acceleration for slip and texture perception. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 145–150, Scottsdale, AZ, May 1989.

[52] T. Iberall, J. Jackson, L. Labbe, and R. Zampano. Knowledge-based prehension: Capturing human dexterity. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 82–87, Philadelphia, PA, April 1988.

[53] T. Iberall and D. Lyons. Towards perceptual robotics. COINS Technical Report 84–17, Laboratory for Perceptual Robotics, Department of Computer and Information Sciences, University of Massachusetts, Amherst, MA, August 1984.

[54] H. Inoue. Manipulation using touch sensors. Technical report, Electrotechnical Laboratory, Tokyo, 1972.

[55] H. Inoue. Tactile pattern recognition for manipulating objects. Technical Note 2, Electrotechnical Laboratory, Tokyo, February 1972.

[56] S. C. Jacobsen, I. D. McCammon, K. B. Biggers, and R. P Phillips. Design of tactile sensing systems for dextrous manipulators. *IEEE Control Systems Magazine*, 8(1):3–13, February 1988.

[57] S. C. Jacobsen, J. E. Wood, D. F. Knutti, and K. B Biggers. The Utah-MIT dextrous hand: Work in progress. *The International Journal of Robotics Research*, 3(4):21–50, 1984.

[58] J. W. Jameson and L. J. Leifer. Quasi-static analysis: A method for predicting grasp stability. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 876–883, San Francisco, CA, April 1986.

[59] R. S. Johannson and A. B. Vallbo. Tactile sensibility in the human hand: Relative and absolute densities of four types of mechanoreceptive units in glabrous skin. *Journal of Physiology*, 286:283–300, 1979.

[60] R. S. Johansson, U. Landstrom, and R. Lundstrom. Responses of mechanoreceptive afferent units in the glabrous skin of the human hand to sinusoidal skin displacements. *Brain Research*, 244:17–25, 1982.

[61] R. S. Johansson and G. Westling. Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental Brain Research*, 56:550–564, 1984.

[62] J. L. Jones and T. Lozano-Pérez. Planning two-fingered grasps for pick and place operations on polydedra. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 683–688, Cincinnati, OH, May 1990.

[63] J. Kerr and B. Roth. Analysis of multifingered hands. *The International Journal of Robotics Research*, 4(4), Winter 1986.

[64] G. I. Kinoshita. Classification of grasped object's shape by an artifical hand with multi-element tactile sensors. In *IFAC Symposium on Information Control Problems in Manufacturing Technology*, pages 111–118, Tokyo, 1977.

[65] G. I. Kinoshita. Representation and tactile sensing of 3-d objects by a gripper finger. *Robotica*, 1:217–222, 1983.

[66] G. I. Kinoshita, S. Aida, and M. Mori. A pattern classification by dynamic tactile sensing information. *Pattern Recognition*, 7:243–251, 1975.

[67] S. Lederman and R. Klatzky. Hand movements: A window into haptic object recognition. In *26th Annual Meeting of the Psychonomic Society*, Boston, MA, 1985.

[68] Z. Li and S. Sastry. Optimal grasping by multifingered robot hands. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 389–394, Raleigh, NC, 1987.

[69] R. C. Luo and W. Tsai. Object recognition using tactile image array sensors. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1248–1253, San Francisco, CA, April 1986.

[70] M. T. Mason. Manipulator grasping and pushing operations. Technical Report 690, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1982.

[71] D. H. McLain. Drawing contours from arbitrary data points. *The Computer Journal*, 17(4):318–324, 1974.

[72] D. J. Montana. *Tactile Sensing and the Kinematics of Contact.* PhD thesis, Harvard University, Cambridge, MA, 1986.

[73] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. *User Guide for Minpack-1*, volume ANL-80-74. Argonne National Laboratory, Argonne, Illinois, August 1980.

[74] S. Narasimhan. Dexterous robotic hands: Kinematics and control. Technical Report 1056, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1988.

[75] S. Narasimhan, D. M. Siegel, and J. M. Hollerbach. Condor: Computational architecture for controlling the Utah-MIT hand. *IEEE Transactions on Robotics and Automation*, 5(5):616–627, October 1989.

[76] T. N. Nguyen and H. E. Stephanou. A topological algorithm for continuous grasp planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 670–675, Cincinnati, OH, May 1990.

[77] V. Nguyen. The synthesis of stable force-closure grasp. Technical Report 905, MIT Artificial Intelligence Laboratory, Cambridge, MA, July 1986.

[78] J. L. Novak. Initial design and analysis of a capacitive sensor for shear and normal force measurement. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 137–144, Scottsdale, AZ, May 1989.

[79] T. Okada and S. Tsuchiya. Object recognition by grasping. *Pattern Recognition*, 9:111–119, 1977.

[80] Y. C. Park and G. P. Starr. Optimal grasping using a multifingered robot hand. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 689–694, Cincinnati, OH, May 1990.

[81] R. P. Paul. *Robot Manipulators.* The MIT Press, Cambridge, MA, 1981.

[82] J. Pertin-Troccaz. On-line automatic programming: A case study in grasping. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1292–1297, Raleigh, NC, 1987.

[83] N. S. Pollard. The grasping problem: Toward task-level programming for an articulated hand. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[84] J. A. Purbrick. A force transducer employing conductive silicone rubber. In *Proceedings of the 1st International Conference on Robot Vision and Sensory Controls*, pages 73–80, Stratford-upon-Avon, UK, 1981.

[85] M. H. Raibert. An all digital VLSI tactile array sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 314–319, Atlanta, GA, 1984.

[86] M. H. Raibert and J. E. Tanner. Design and implementation of a VLSI tactile sensing computer. *The International Journal of Robotics Research*, 1(3):3–18, 1982.

[87] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, 1986.

[88] R. A. Russell. A thermal sensor array to provide tactile feedback for robots. *The International Journal of Robotics Research*, 5(3):35–39, 1985.

[89] M. A. Sabin. Contouring — a review of methods for scattered data. In K. W. Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, pages 63–86. Academic Press, New York, NY, 1980.

[90] J. K. Salisbury. Kinematic and force anaysis of articulated hands. Department of Computer Science STAN-CS-82-921, Stanford University, Stanford, CA, 1982.

[91] J. L. Schneiter. An objective tactile sensing strategy for object recognition and localization. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1262–1267, San Francisco, CA, April 1986.

[92] J. L. Schneiter and T. B. Sheridan. An optical tactile sensor for manipulators. *Robotics and Computer-Integrated Manufacturing*, 1(1):65–71, 1984.

[93] D. Shepard. A two-dimensional function for irregularly spaced data. In *Proceedings of 23rd ACM National Conference*, pages 517–524, 1968.

[94] D. M. Siegel, I. Garabieta, and J. M. Hollerbach. A capacitive based tactile sensor. In *Proceedings of SPIE Conference on Intelligent Robots and Computer Vision*, Cambridge, MA, September 1985.

[95] D. M. Siegel, I. Garabieta, and J. M. Hollerbach. An integrated tactile and thermal sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1286–1291, San Fransico, CA, April 1986.

[96] D. M. Siegel and L. Simmons. A thermal based sensor system. In *Proceedings of SME Sensors '85 Conference*, Detroit, MI, November 1985.

[97] S. S. Skiena. Geometric probing. Department of Computer Science UILU-ENG-88-1730, University of Illinois at Urbana-Champaign, Urbana, Illinois, April 1988.

[98] W. E. Snyder and J. St. Clair. Conductive elastomers as sensor for industrial parts handling equipment. *IEEE Transactions on Instrumentation and Measurement*, IM-27:94–99, 1978.

[99] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[100] S. A. Stansfield. Robotic grasping of unknown objects: A knowledge-based approach. Report SAND89-1087, Sandia National Laboratory, Albuquerque, NM, June 1989.

[101] C. J. Stone. Nearest neighbor estimators of a nonlinear regression. In *Proceedings of Computer Science and Statistics: 8th Annual Symposium on the Interface*, pages 413–418, 1975.

[102] B. Tise. A compact high resolution piezoresistive digital tactile sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 760–764, Philadelphia, PA, April 1988.

[103] R. Tomovic, G. Bekey, and W. Karplus. A strategy for grasp synthesis with multifingered robot hands. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 83–89, Raleigh, NC, 1987.

[104] R. Tomovic and G. Boni. An adaptive artificial hand. *IRE Transactions on Automatic Control*, AC-7(3), April 1962.

[105] M. J. Usher. *Sensors and Transducers.* Macmillan, London, 1985.

[106] A. B. Vallbo and R. S. Johansson. The tactile sensory innervation of the glaborous skin of the human hand. In G. Gordon, editor, *Active Touch: The Mechanism of Recognition of Objects by Manipulation*, pages 26–54. Pergamon Press, New York, NY, 1978.

[107] G. S. Watson. Smooth regression analysis. *The Indian Journal of Statistics, Series A*, 26:359–372, 1964.

[108] Wind River Systems, Inc., 1351 Ocean Ave., Emeryville, CA 94608. *VxWorks Programmer's Guide*, 1989.